DIE ONTWIKKELING VAN 'N 8-PSS TRELLISKODE DATASENDER EN VITERBI-ONTVANGER

P.E. HERTZOG

DIE ONTWIKKELING VAN 'N 8-PSS TRELLISKODE DATASENDER EN VITERBI-ONTVANGER.

deur

PIERRE EDUARD HERTZOG

Verhandeling voorgelê ter voldoening aan die vereistes vir die

MAGISTER TECHNOLOGIAE
INGENIEURSWESE : ELEKTRIES

in die

Fakulteit Ingenieurswese

aan die

Technikon Vrystaat

1999

Studieleier : Prof. G.D. Jordaan D.Tech (Ing)

Central University of Technology, Free State

VERKLARING TEN OPSIGTE VAN SELFSTANDIGE WERK

Ek, PIERRE EDUARD HERTZOG, verklaar hiermee dat die navorsingsprojek wat vir die verwerwing van die graad MAGISTER TECHNOLOGIAE: INGENIEURSWESE: ELEKTRIES aan die Technikon Vrystaat deur my voorgelê word, my selfstandige werk is en nie voorheen deur my of enige ander persoon ter verwerwing van enige kwalifikasie voorgelê is nie.

18/3/99

HANDTEKENING VAN STUDENT DATUM

BEDANKINGS

Graag bedank ek die volgende persone en instansies vir hulp en ondersteuning waarsonder hierdie projek nie suksesvol voltooi kon word nie.

My eggenote, Sunette, my seuns Dirkie en Hansie, sowel as my en ook Sunette se ouers, vir ondersteuning, bemoediging en tyd aan my gegun.

My kollegas vir hulp en ondersteuning, veral Johan wat vir my tot groot hulp was in voltooiing van die projek.

Die Stigting vir Navorsingsontwikkeling (SNO) vir finansiële hulp.

Technikon Vrystaat vir die geleentheid om die projek te voltooi.

Aan my studieleier, Prof. G.D. Jordaan, wat 'n baie groot aandeel het in die voltooiing van hierdie projek en waarvoor ek die grootste respek het.

Laastens aan my Hemelse Vader waarsonder niks moontlik is nie.

OPSOMMING

Die ontvangs van geldige (foutvrye) digitale data na transmissie oor 'n kanaal is van die grootste belang in die moderne samelewing.

As deel van die projek is 'n trelliskode datasender, sowel as Viterbi-ontvanger, ontwikkel en ge-evalueer. Die projek het die gebruik van intelligente, DSP-gebaseerde hardeware en die ontwikkeling van die gepaardgaande sagteware behels. Die viterbi algoritme is as ontwerpfilosofie van die ontvanger aanvaar a.g.v. die vermoë daarvan om foutief ontvangde data te identifiseer en deels te korrigeer. Die realisering van hierdie tegniek noodsaak die gebruik van 'n prosesseerder met 'n baie hoë verwerkingspoed. Om hierdie rede is die DSP56001 digitale sein prosessering (DSP) mikroprosesseerder uiters geskik vir implementering in hierdie stelsel.

Die implementering van die stelsel is voorafgegaan deur rekenaar simulasies ten einde die verwagte werkverrigting van die stelsel te bepaal.

Enige kommunikasiekanaal het 'n unieke bandwydte. Probleme ontstaan egter wanneer gepoog word om hoëspoed datakommunikasie oor kanale met 'n beperkte bandwydte te bewerkstellig a.g.v. die verband tussen datatempo en die benodigde bandwydte. Hoe smaller die bandwydte is, hoe minder data kan per eenheidstydsduur versend word. Tydens die uitvoering van die projek is trelliskode gemoduleerde data oor 'n kanaal met 'n beperkte bandwydte teen verskillende seinruisverhoudings versend en die werking van die sender en die ontvanger bepaal.

SUMMARY

The reception of valid (error free) digital data after transmission is of the utmost importance in the modern society.

As part of the project a trellis code data transmitter, as well as a Viterbi receiver was developed and evaluated. The project entails the use of intelligent DSP hardware and the development of associated software. The Viterbi algorithm was accepted as the design philosophy of the receiver because of its ability to identify erroneous data received and to correct it if possible. The realization of this technique necessitates the use of a processor with very high processing speed. For this reason the DSP56001 digital signal processor (DSP) was chosen as it is admirably suited for implementation in this system.

The implementation of the system was preceded by computer simulations to determine the expected performance of the system.

Any communication channel has a unique bandwidth. Problems however, occur when high-speed data communication is attempted over channels with limited bandwidth because of the relation between data rate and the required bandwidth. The narrower the bandwidth, the less data can be transmitted per unit time.

During the execution of the project, trellis code modulated data was transmitted via a channel with limited bandwidth at different signal to noise ratios and the performance of the transmitter and receiver determined.

Central University of Technology, Free State

AFKORTINGS

APSS	Amplitude Fase Skuifsleuteling
BER	Bisfouttempo
BPRZ	Bipolêr-herstel-na-zero
BPSS	Binêre fase skuifsleuteling
bps	Bisse per sekonde
DSP	Digitale seinprosessering
FSS	Frekwensie skuifsleuteling
LDF	Laagdeurlaatfilter
NRZ	Nie-herstel-na zero
PSS	Fase skuifsleuteling
RZ	Herstel-na-zero

Central University of Technology, Free State

INHOUD

Central University of Technology, Free State

LYS VAN FIGURE

Central University of Technology, Free State

LYS VAN TABELLE

1 Inleiding

Gedurende die tweede helfte van die huidige eeu was daar 'n groot toename in die gewildheid van digitale kommunikasie. Dit kan grotendeels toegeskryf word aan die volgende faktore:

- ✓ Die tegnologie op die gebied van digitale kommunikasie het oor die afgelope paar jaar geweldig ontwikkel.
- ✓ Digitale seinprosessering (DSP) het moontlik en bekostigbaar geraak vir individue wat navorsing en ontwikkeling in hierdie gebied doen.
- ✓ Die behoefte aan datanetwerke wat terselfdertyd meer as een gebruiker teen verskillende datatempo's moet bedien.
- ✓ Die implementering van satellietstelsels wat teen hoë datatempo's moet funksioneer as gevolg van hoë lanseringskostes.
- ✓ Groot rekenaar databanke wat verskeie gebruikers bedien.
- ✓ Beter benutting van die beskikbare frekwensieband.
- ✓ Meer betroubare stelsels.
- ✓ Die vermoë van hedendaagse stelsels om 'n verswakte of verwronge digitale sein weer tot die oorspronklike te herstel.

Indien bogenoemde in ag geneem word kan die belangrikheid van navorsing in die gebied van digitale kommunikasie nie oorbeklemtoon word nie.

In hierdie projek is die gebruik van trelliskodemodulasie en viterbi-dekodering ondersoek. Daar is van digitale seinprosesseringstegnieke gebruik gemaak in die implementering van 'n 8-PSS trellis ge-enkodeerde datasender en viterbi-ontvanger. Die voordele van hierdie tegnieke bo konvensionele analoog ontwerpe is as volg:

- ✓ Faktore soos temperatuurveranderinge, wat 'n rol by analoogstelsels speel, word hiermee uitgeskakel. Data kan dus met baie groter presiesheid en akkuraatheid verwerk word.

- ✓ As gevolg van die presisie waarmee seine geprosesseer word is die resultate volkome herhaalbaar en voorspelbaar.

- ✓ Daar word grootliks van sagteware gebruik gemaak vir die ontwikkeling van die stelsel. Veranderinge en verbeteringe kan dus sonder veel moeite en addisionele koste aangebring word.

- ✓ Die ontwerp kan in sagteware gesimuleer, ontfout en geëvalueer word voor implementering.

- ✓ Sekerlik die grootste voordeel van die DSP stelsel is die feit dat die kompleksiteit van die stelsel haas onafhanklik is van die hardeware [1, p. 3].

- ✓ Meeste DSP sisteme maak gebruik van Harvard argitektuur. Dit behels dat die programgeheue en die datageheue op verskillende busse en onafhanklik

van mekaar kan funksioneer. Die voordeel hiervan is dat die data en die volgende programinstruksie terselfdertyd gehaal kan word [1, p. 8].

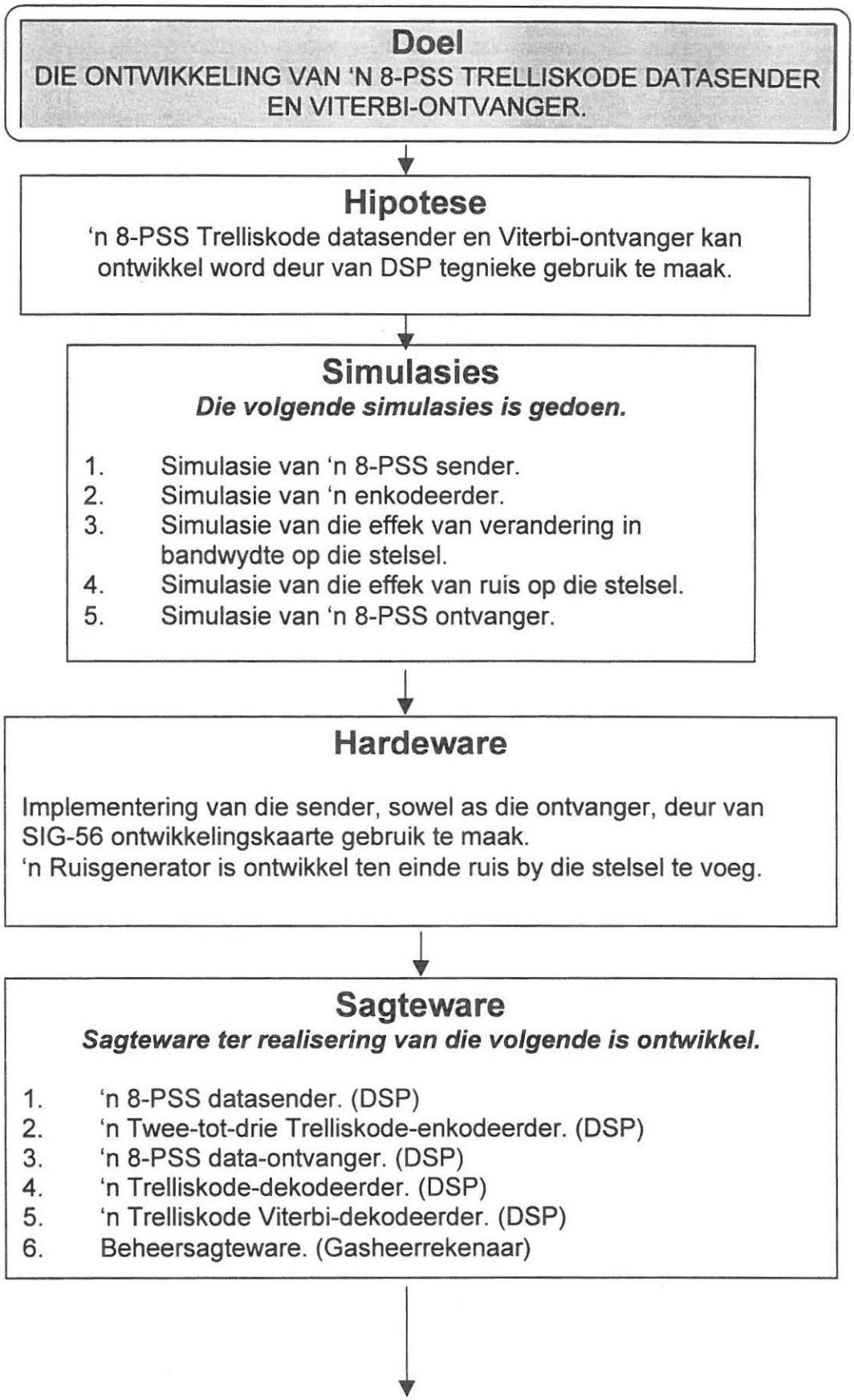
1.1 Doelwitte van die projek

Die volgende doelwitte is vir die projek gestel:

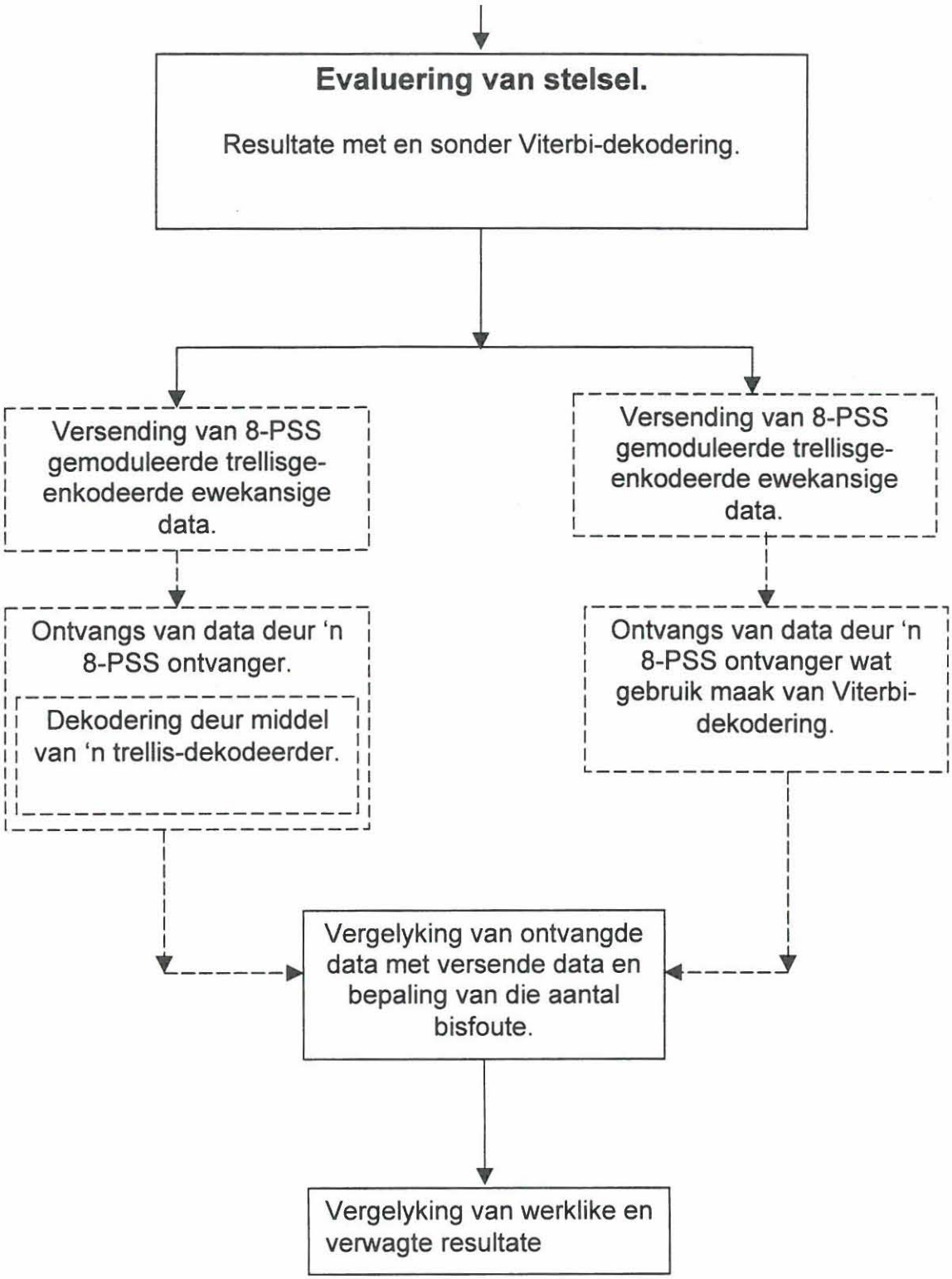
- Die bestudering van trelliskodemodulasie sowel as viterbi-dekodering.
- Die simulering en ontwikkeling van 'n trellis-enkodeerder en 'n 8-PSS datasender.
- Die ontwikkeling van 'n 8-PSS ontvanger wat gebruik maak van viterbi-dekodering.
- Die implementering van die sender sowel as die ontvanger in DSP deur van SIG-56 DSP ontwikkelingskaarte gebruik te maak.
- Evaluering van trelliskodemodulasie en die DSP implementering daarvan.

1.2 Verloop van projek

Die verloop van die projek word vervolgens blokdiagramaties voorgestel.



Figuur 1-1: Blokdiagrammatiese voorstelling van die verloop van die projek.



Figuur 1-1: (Vervolg) Blokdiagrammatiese voorstelling van die verloop van die projek.

2 Datakommunikasie

2.1 Inleiding

Die doel van 'n kommunikasiestelsel is die oordrag van inligting vanaf die bron na die verbruiker op die doeltreffendste moontlike manier.

2.2 Enkodering van digitale seine

Digitale seine is nie altyd geskik vir transmissie as basisbandseine nie. Dit is dikwels nodig om die bisse eers op een of ander wyse te manipuleer voor transmissie. Hier volg 'n aantal probleme wat moontlik kan voorkom indien databisse nie voor versending gemanipuleer word nie.

- Wanneer daar 'n aantal nulle of ene direk na mekaar versend word, gebeur dit dat 'n konstante spanningsvlak vir 'n periode op die transmissielyn teenwoordig is. Die probleem word gelykstroomskuif (DC shift) genoem [16, p. 140]. Hierdie is 'n probleem omdat baie transmissiestelsels nie voorsiening maak vir versending van 'n gelykstroomkomponent nie.

- 'n Verdere probleem is dat die klok van die sender en dié van die ontvanger, presies in pas moet wees. Selfs 'n klein afwyking kan veroorsaak dat daar 'n groot aantal datafoute voorkom. By die ontvanger word die klok afgelei van die inkomende datastroom. Indien daar dus vir 'n lang periode slegs nulle, of slegs ene, ontvang word is dit baie moeilik om die klok te herwin.

Die oplossing vir albei hierdie probleme is enkodering van die data. Enkodering impliseer die manipulering van die oorspronklike datastroom op so 'n wyse dat daar 'n konstante patroon in die datastroom voorkom. By die ontvangkant is daar 'n dekodeerder wat die proses omkeer en herwinning van die oorspronklike data moontlik maak.

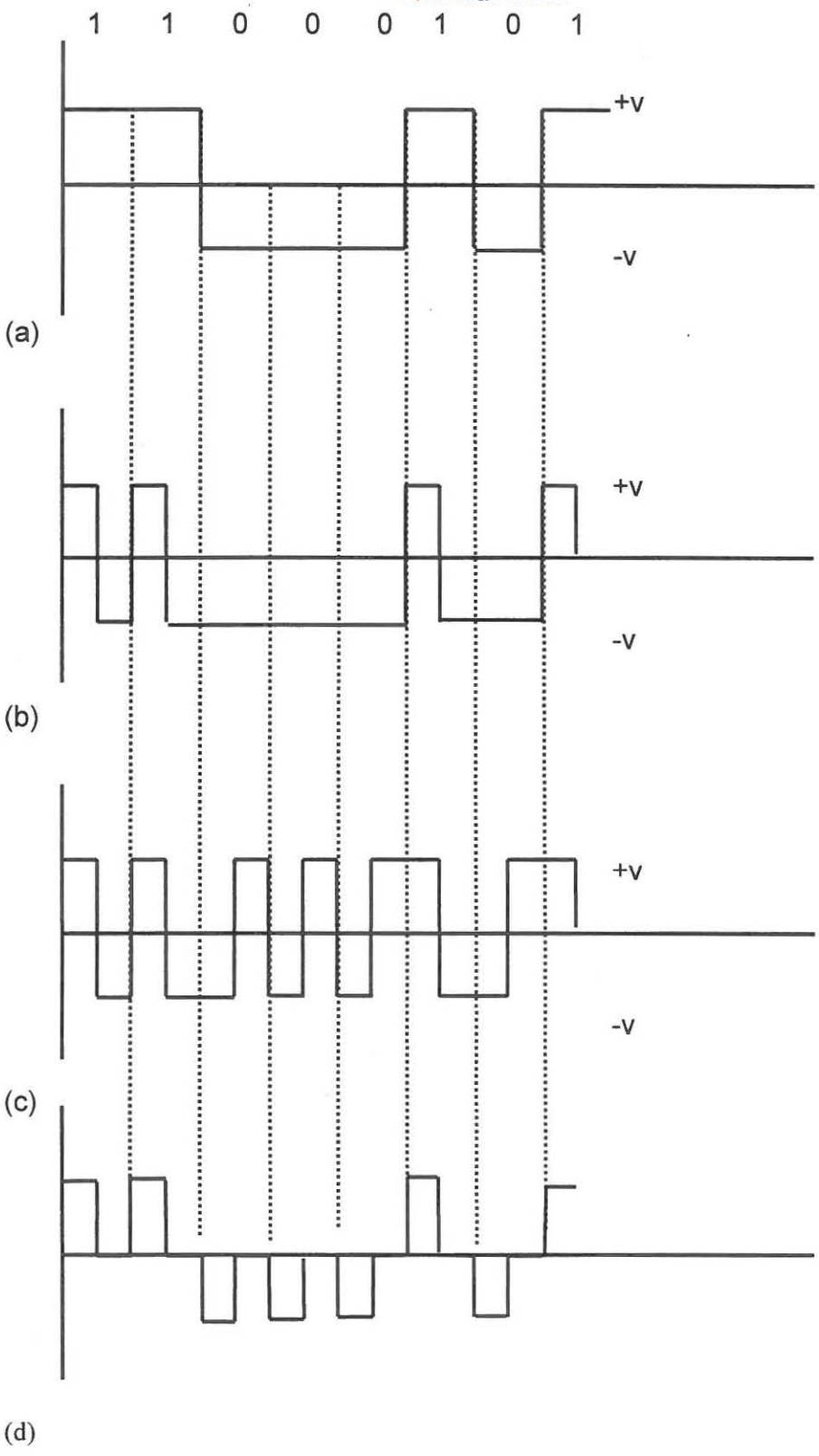
Vervolgens word 'n paar enkoderingstegnieke beskryf.

2.2.1 Nie-herstel-na-zero (NRZ)

By die NRZ stelsel word die digitale data net soos dit is, versend. Dit is dus onge-enkodeer. Wanneer die bisse 11000101 byvoorbeeld gestuur word, sal die versende golf dieselfde wees, dus 11000101. Die data word dus nie gemanipuleer nie. Figuur 2-1 (a) toon die nie-herstel-na-zero sein vir 'n kort datastroom.

2.2.2 Herstel-na-zero (RZ)

By die RZ stelsel word die bistyd in twee gedeel. Gedurende die eerste helfte van die bistyd is die sein dieselfde as die NRZ stelsel, maar gedurende die tweede helfte van die bistyd word daar na nul teruggekeer. Wanneer daar opeenvolgens 'n klomp ene gestuur word, sal die uitset dus 'n alternerende sein wees, wat wissel tussen een en nul. Figuur 2-1 (b) toon die herstel-na-zero sein vir 'n kort datastroom. 'n Opeenvolgende aantal nulle veroorsaak steeds 'n gelykstroom komponent.



Figuur 2-1: Tipiese enkoderingstegnieke: a)NRZ, b)RZ, c)Manchester-enkodering en d)BPRZ.

2.2.3 Manchester-enkodering

By Manchester-enkodering het elke bis 'n hoog sowel as 'n laag. Wanneer 'n een versend word is die eerste deel van die bisperiode dus 'n een en gedurende die tweede helfte 'n nul. In die geval waar 'n nul versend word sal die spanningsvlak vir die eerste helfte van die bistyd 'n nul wees, maar gedurende die tweede helfte sal daar na 'n een geskakel word. Elke bistyd bestaan dus uit 'n een sowel as 'n nul. Dit vergemaklik sinchronisasie. Figuur 2-1 (c) toon Manchester-enkodering vir 'n kort datastroom.

2.2.4 Bipolêr-herstel-na-zero (BPRZ)

Indien BPRZ gebruik word, en 'n een versend word, sal daar 'n positiewe spanningsvlak vir die eerste helfte van die bistyd teenwoordig wees, en gedurende die tweede helfte van die bistyd word na 'n neutrale posisie tussen +V en -V beweeg. Indien daar egter 'n nul versend word sal die eerste helfte van die bistyd 'n negatiewe vlak hê terwyl die tweede helfte 'n neutrale vlak sal hê [20, p. 201]. Figuur 2-1(d) toon bipolêr-herstel-na-zero enkodering vir 'n kort datastroom.

2.3 Inligtingskapasiteit van 'n kommunikasiekanaal

Die hoof parameters in 'n kommunikasiestelsel is werkverrigting, bandwydte en kompleksiteit of koste. Indien 'n kommunikasiestelsel vir 'n spesifieke toepassing ge-evalueer word, moet al bogenoemde punte in ag geneem word. By die evaluering van 'n analoogstelsel word daar gewoonlik gekyk na die

seinruisverhouding op die uitset van die stelsel. By 'n digitale stelsel word gelet op die waarskynlikheid van 'n bisfout [3, p. 4].

'n Gemeenskaplike doel van kommunikasiestelselteorie is om limiete te definieer betreffende die werkverrigting van 'n stelsel. Wanneer sulke limiete vir 'n stelsel bekend is, kan besluit word of die stelsel naby genoeg aan die optimale teoretiese waardes werk, en of die moontlikheid van verbetering addisionele ontwikkelingskostes regverdig [3, p. 6].

Die inligtingskapasiteit van 'n kommunikasiestelsel stel die aantal onafhanklike simbole wat binne 'n sekere tyd gestuur kan word voor. Die mees basiese simbool is die binêre bis. Om hierdie rede word daar dikwels verwys na inligtingskapasiteit in terme van bisse per sekonde (bps). Die inligtingskapasiteit van 'n stelsel is direk proporsioneel aan die bandwydte en transmissietyd [20, p. 3].

Shannon het die limiet vir inligtingskapasiteit gedefinieer. Volgens sy limiet kan 'n redelike klein aantal bisfoute verkry word indien binne die berekende kanaalkapasiteit van 'n bepaalde stelsel gewerk word. Volgens Shannon word die kanaalkapasiteit as volg bereken [14, p. 159]:

$$C = B\log_2(1+\frac{S}{N}) \qquad (2\text{-}1)$$

Waar:

C = *inligtingskapasiteit (bps)*

B = bandwydte van kanaal (in Hz)

$\frac{S}{N}$ = seinruisverhouding

Die implikasies van bostaande vergelyking is dat dit teoreties moontlik is om oor 'n kanaal met 'n bandwydte van B, teen 'n bistempo van kleiner of gelyk aan C, te werk sonder noemenswaardige bisfoute. Verder wys Shannon se formule daarop dat dit wel moontlik is om op 'n raserige kanaal met 'n spesifieke seinruisverhouding, teen 'n lae bistempo effektief te kan werk.

2.4 Digitale modulasietegnieke

'n Paar digitale modulasietegnieke word vervolgens bespreek. Die klem word op 8-PSS gelê omdat dit die modulasietegniek is wat in die projek gebruik is.

2.4.1 Frekwensie skuifsleuteling (FSS)

By FSS word die draer se frekwensie geskuif met 'n verandering in insetdata. Vir elke datavlak word daar 'n spesifieke uitsetfrekwensie deur die modulator gegenereer. Die uitset kan byvoorbeeld 1000Hz wees indien 'n 1 op die inset van die modulator geplaas word, en 1200Hz indien 'n 0 op die inset teenwoordig is.

Volgens Sklar is die uitdrukking vir FSS [17, p. 130]:

$$S_i(t) = \sqrt{\frac{2E}{T}}\cos[\omega_i t + \phi] \quad (2\text{-}2)$$
$$0 \le t \le T$$
$$i = 1,\ldots,M$$

Waar: ω_i = frekwensiekomponent wat uit M aantal diskrete waardes bestaan.

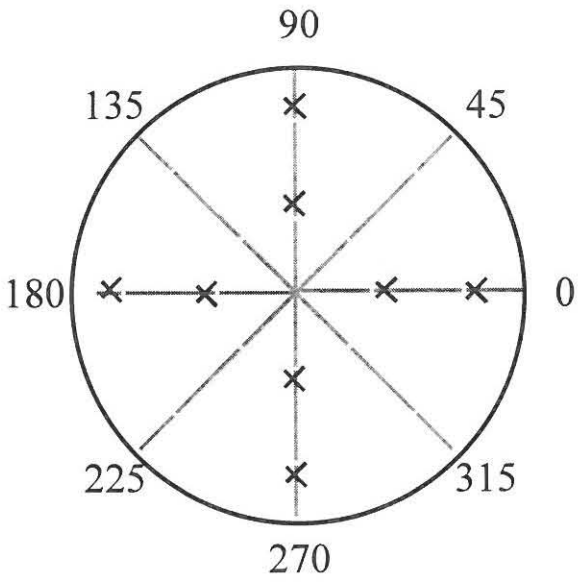
ϕ = fasekomponent wat konstant is.

E = simboolenergie.

T = die periode.

2.4.2 Amplitude Fase Skuifsleuteling (APSS)

By APSS verander nie net die fase nie maar ook die amplitude van die sein. Figuur 2-2 dui 'n konstellasiediagram waar nie net die fase verander nie, maar ook die amplitude. Vier fases kan waargeneem word en elk van hierdie vier fases kan twee amplitude vlakke aanneem. Dus word daar 8 moontlike simbole in die diagram getoon.



Figuur 2-2: 'n Tipiese konstellasiediagram van amplitude fase skuifsleuteling.

Die algemene analitiese uitdrukking vir APSS is [17, p. 131]:

$$S_i(t) = \sqrt{\frac{2E_i(t)}{T}}\cos(\omega_o t + \phi_i(t)) \qquad (2\text{-}3)$$

$$\text{Die amplitude term} = \sqrt{\frac{2E_i(t)}{T}}$$

Waar: ω_o = frekwensiekomponent wat konstant is.

ϕ = fasekomponent wat uit M aantal diskrete waardes bestaan.

E = simboolenergie.

T = die periode.

2.4.3 Fase skuifsleuteling (PSS)

8-Fase skuifsleuteling is die modulasietegniek wat in die projek gebruik is en gevolglik word PSS meer breedvoerig ondersoek. PSS is soortgelyk aan gewone fase modulasie vir analoogseine. Die enigste verskil is dat daar slegs 'n vasgestelde aantal faseveranderings kan voorkom. Hierdie aantal faseveranderings is afhanklik van die aantal binêre vlakke wat op die inset van die stelsel moontlik is. Volgens Sklar is die analitiese uitdrukking vir PSS [17, p. 130]:

$$S_i(t) = \sqrt{\frac{2E}{T}}\cos[\omega_o t + \phi_i(t)]$$
$$0 \le t \le T \qquad (2\text{-}4)$$
$$i = 1,\ldots,M$$

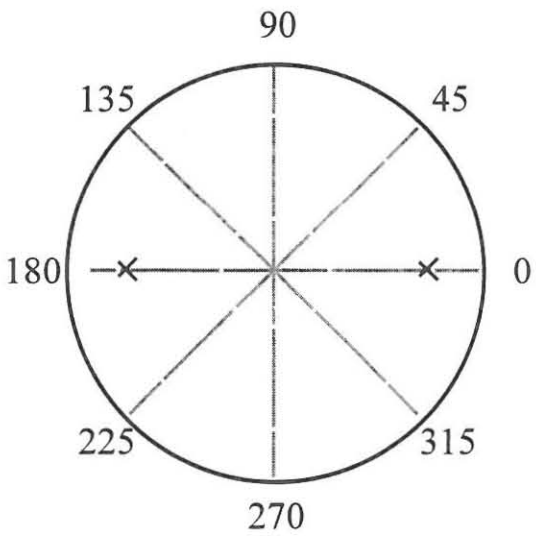
Waar: ω_o = frekwensiekomponent wat konstant is.

ϕ = fasekomponent wat uit M aantal diskrete waardes bestaan.

E = simboolenergie.

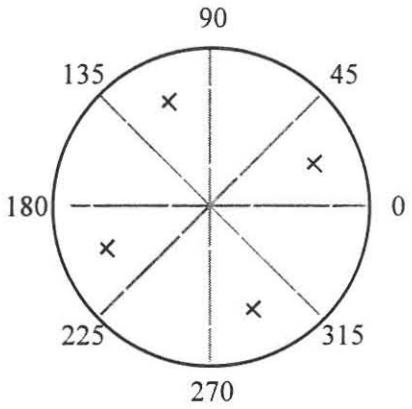
T = die periode.

By binêre fase skuifsleuteling (BPSS) is daar twee moontlike uitsetfases. Die een fase verteenwoordig 'n 1 en die ander verteenwoordig 'n 0. Die uitset van die modulator skuif dus tussen twee fases wat 180° van mekaar verwyder is. Figuur 2-3 illustreer die uitsetfases van so 'n stelsel.



Figuur 2-3: Moontlike konstellasie van 'n binêre fase skuifsleuteling sein.

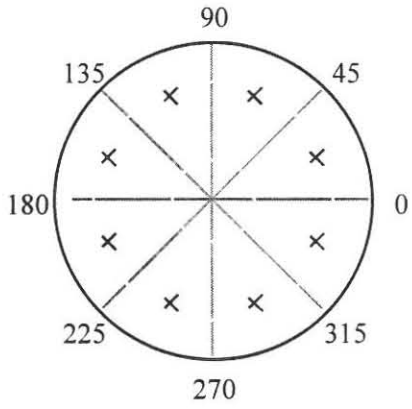
In kwadratuur fase skuifsleuteling (KPSS) is daar vier moontlike uitset fases. Die inset tot die stelsel kan dus 00, 01, 10 of 11 wees. Die vier uitsetfases is 90° van mekaar geposisioneer soos aangetoon in Figuur 2-4.



Figuur 2-4: Moontlike konstellasie van 'n kwadratuur fase skuifsleutelingsein.

By 'n 8-Fase skuifsleuteling (8-PSS) modulator is daar 8 verskillende uitsetfases. Ten einde 8 fases te enkodeer, word daar 3 bisse ('n tribis) op 'n slag op die enkodeerder inset gekoppel ($2^3 = 8$).

Figuur 2-5 dui die agt uitsette van 'n tipiese 8-PSS stelsel aan.



Figuur 2-5: Moontlike konstellasie van die uitset fases van 'n 8-PSS sender.

Die waarskynlikheid van foute vermeerder met 'n vermeerdering in die aantal fases van 'n PSS stelsel. Indien daar meer moontlike fases is, beteken dit dat hulle nader aan mekaar op die fasediagram (konstellasie) voorkom en dus is die moontlikheid dat 'n fasefout tydens ontvangs kan voorkom, groter. Die teoretiese werkverrigting van die PSS stelsel word deur die volgende uitdrukking weergegee [4, p. 317].

$$Pe = erfc\left(\sqrt{\frac{E}{N_o}}\sin\left(\frac{\pi}{M}\right)\right) \qquad (2\text{-}5)$$

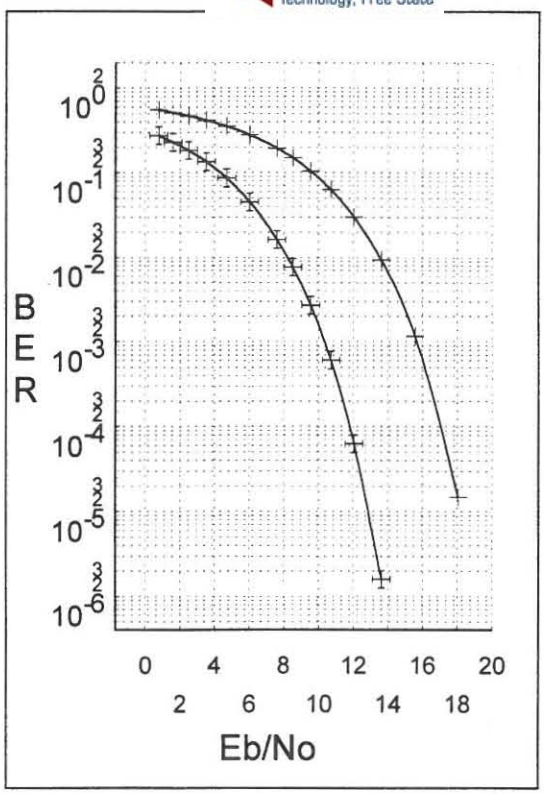
Waar: Pe = die waarskynlikheid van 'n simboolfout.

M = die aantal fases van die PSS stelsel.

E = die seinenergie per simbool.

N_o = die ruisenergie.

Figuur 2-6 *toon* die verwagte bisfouttempo vir 8-PSS en 4-PSS teen verskillende seinruisverhoudings [4, p. 336]. In die skets kan gesien word hoe die waarskynlikheid van foute vermeerder indien van 'n 4 na 'n 8-PSS stelsel beweeg word.



Figuur 2-6: Die teoreties berekende BER van 8-PSS en 4-PSS seine teen verskillende seinruisverhoudings [4, p. 336].

2.5 Opsomming

Daar bestaan 'n groot aantal digitale modulasie- en enkoderingstegnieke wat algemeen toegepas word. Die keuse van watter tegniek gebruik gaan word hang grotendeels van die toepassing sowel as die beskikbare bandwydte af. Die enkodering van digitale seine het verskeie voordele waarvan die vermoë om foutiewe data te identifiseer, en in sommige gevalle te korrigeer, sekerlik die grootste is.

3 Digitale enkoderingstelsels

Daar is twee koderingstegnieke wat algemeen in datakommunikasie gebruik word, naamlik blokkodes en konvolusiekodes. In die projek word van trelliskode gebruik gemaak en gevolglik word konvolusiekodes breedvoeriger behandel.

3.1 Blokkodes

By blokkodes word die inkomende datastroom in blokke opgedeel. Elk van hierdie blokke bestaan uit 'n sekere aantal bisse. Gestel 'n blok het k inligtingsbisse, dan is daar 2^k moontlike binêre woorde. Daar word egter addisionele bisse bygevoeg deur die enkodeerder. Hierdie addisionele bisse word gebruik om foute na ontvangs van die sein te identifiseer en moontlik reg te stel. Sodoende word die invloed van ruis op die stelsel beperk [2, p. 3].

3.1.1 Die byvoeging van 'n pariteitsbis

Die eenvoudigste blokkode voeg 'n enkel bis by die k aantal databisse van elke datawoord. Hierdie bygevoegde bis word 'n pariteitsbis genoem, en word bygevoeg om die totale aantal ene in die woord ewe of onewe, afhangend van die stelsel, te maak. Foute in 'n ewe aantal bisposisies kan egter nie deur pariteitstoetsing herken word nie. In baie stelsels is die moontlikheid van meer as een fout per woord egter onwaarskynlik. Wanneer daar van 'n enkele pariteitsbis gebruik gemaak word is dit nie moontlik om foutregstelling te doen nie. Deur die byvoeging van meer toetsbisse word die vermoë om regstellings te doen egter groter [11, p. 89]. Die Hammingkode bestaan uit 'n matriks

waarop pariteitstoetsing gedoen word ten einde moontlike foute te identifiseer en ook reg te stel [11, p. 93].

3.2 Konvolusiekodes

By konvolusiekodes word daar nie soos by blokkodes van afsonderlike blokke data gebruik gemaak nie. Die enkodering is 'n aaneenlopende proses, waar 'n kontinue stroom data aan die enkodeerder gevoer word. By konvolusiekodes voeg die konvolusie-enkodeerder op 'n deurlopende basis addisionele bisse by die inkomende databisse. Indien daar twee bisse op die uitset van die enkodeerder verskyn vir elke een inset bis word dit 'n 1-tot-2 enkodeerder genoem. Konvolusie enkodeerders het 'n tipe geheue. Dit beteken dat vorige data 'n invloed het op die besluitneming wat op die huidige stadium deur die enkodeerder gedoen word [14, p. 183].

3.2.1 Trelliskodemodulasie

Hoe groter die bandwydte, hoe makliker is dit om die invloed van ruis te beperk en 'n betroubare stelsel te verseker. In baie gevalle moet die modems so ontwerp word dat hulle op normale telefoonlyne met 'n spesifieke bandwydte werk. Trelliskodemodulasie het die voordeel dat dit 'n hoë kodewins, sonder die toename in bandwydte het, indien dit met 'n ongekodeerde modulasiestelsel vergelyk word [8, p. 330].

Die werkverrigting van 'n ongekodeerde PSS sein hang af van die afstand tussen twee aangrensende fases en die gemiddelde seindrywing van die sender [17, p. 418]. Die afstand tussen opeenvolgende fases verminder met 'n

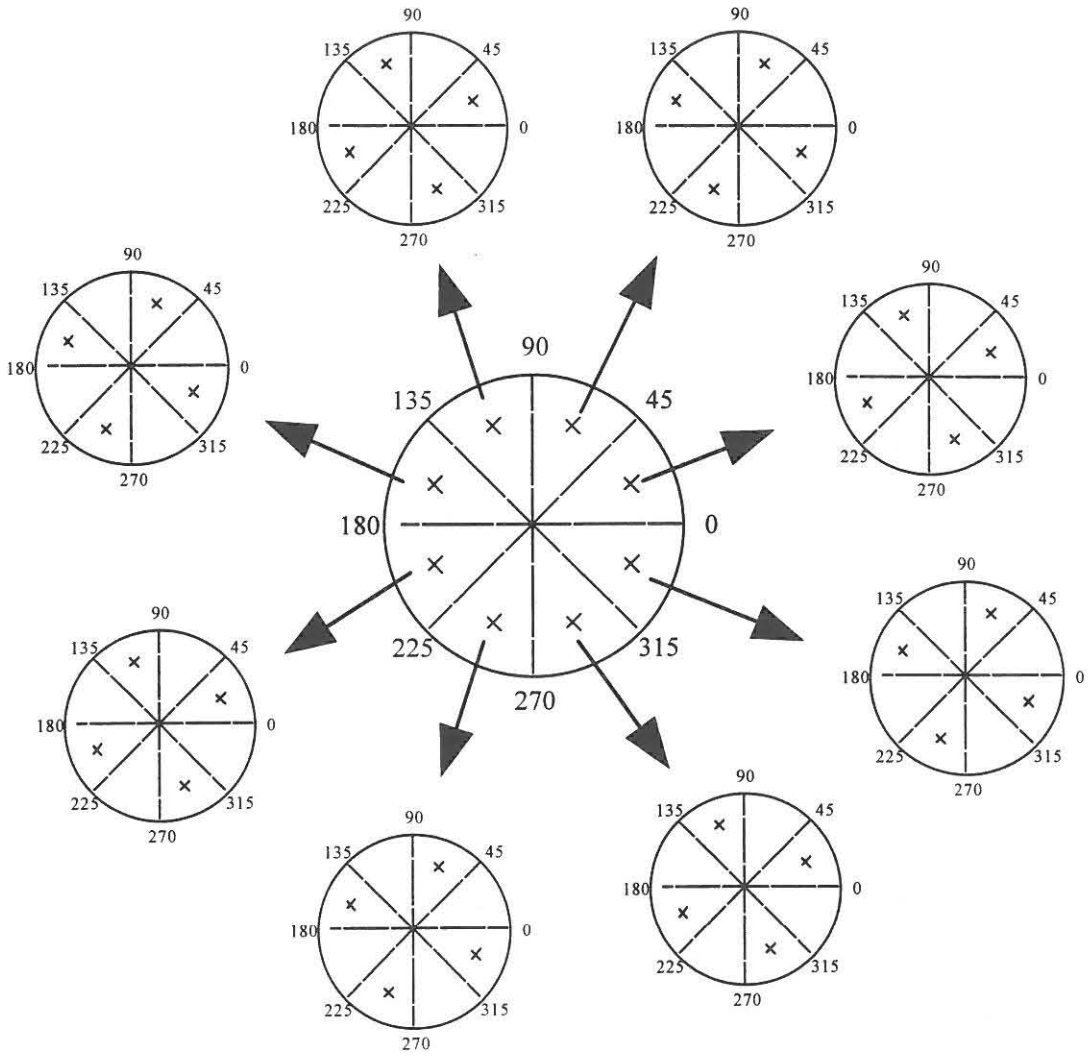
toename in die aantal fases. Die doel van trelliskodering is om die afstand tussen die seine wat mees waarskynlik verwar kan word, te vergroot, sonder om die gemiddelde seindrywing te vermeerder. Deur van trelliskodes gebruik te maak kan effektief vanaf 'n 4-PSS stelsel na 'n 8-PSS stelsel oorgeskakel word, sonder om die bandwydte, of die waarskynlikheid van foute, te verhoog.

Ter verduideliking word van 'n trellis ge-enkodeerde 8-PSS stelsel as 'n voorbeeld gebruik gemaak. Vanaf enige posisie in die 8-PSS stelsel kan slegs na vier posisies ten opsigte van die volgende tribit beweeg word. Die moontlikheid van 'n fout word beperk deurdat in plaas van 45 grade spasiëring tussen moontlike ontvangde data, daar nou 90 grade spasiëring is. In Figuur 3-1 kan gesien word dat daar van enige van die agt fases slegs na vier, en nie agt, moontlike posisies gespring kan word nie. Indien die data nie ge-enkodeer was nie sou agt posisies moontlik gewees het.

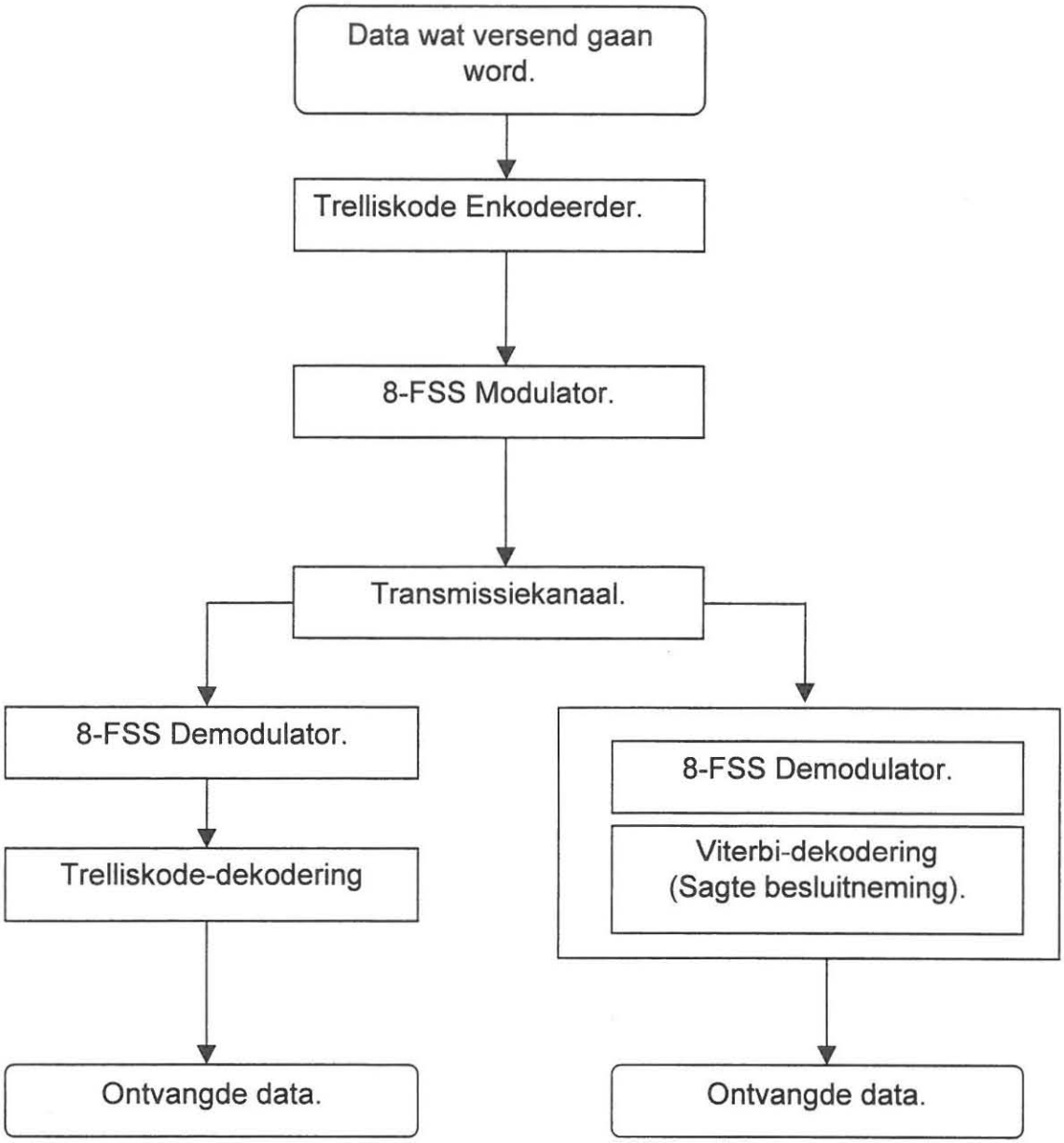
3.2.2 'n Tipiese trelliskodestelsel

Ten einde trelliskodemodulasie te ondersoek is simulasies van tipiese stelsels gedoen. Hierdie stelsels, wat uit die volgende bestaan, en in Figuur 3-2 uiteengesit is, word vervolgens bespreek.

- 'n 2-tot-3 Enkodeerder.
- 'n 8-PSS modulator.
- 'n 8-PSS demodulator.
- 'n Viterbi dekodeerder.
- 'n Trelliskode-demodulator.



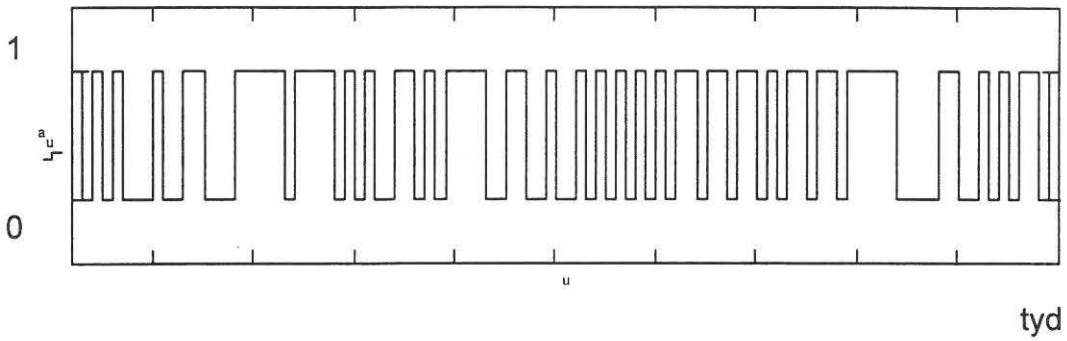
Figuur 3-1 Aanduiding van moontlike spronge van ge-enkodeerde 8-PSS.



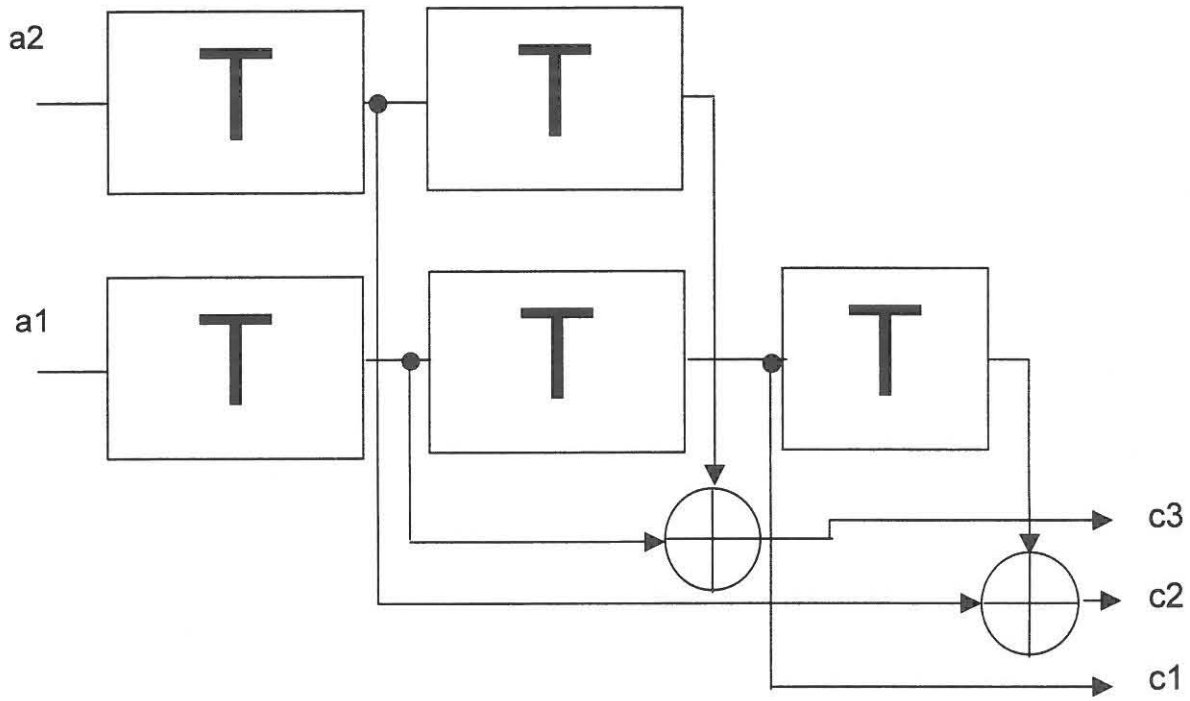
Figuur 3-2: Blokdiagram van 'n gesimuleerde trelliskode sender en twee tipes ontvangers.

3.2.3 Bespreking van die enkodeerder.

In die simulasie moes daar eerstens 'n ewekansige datastroom gegenereer word. Hierdie datastroom is as inset van die enkodeerder gebruik.



Figuur 3-3: Ewekansige datastroom soos gegenereer in MathCAD[1]



Figuur 3-4: 2-tot-3 Enkodeerder.

Die inkomende datastroom word op die inset van die enkodeerder in Figuur 3-4 toegepas. Die eerste databis word op a1 (minsbeduidend) toegepas en die tweede op a2 (meesbeduidend). Wanneer a1 en a2 op die inset teenwoordig is word dit gelyktydig na die uitset van die eerste houkringe, aangedui deur T,

verplaas. Die enkodeerder sal dan 'n uitset gee en gereed wees om die volgende twee databisse te ontvang. By die uitset van die enkodeerder is c1 die minsbeduidende bis en c3 die meesbeduidende bis.

Die datastroom word op hierdie wyse ge-enkodeer. Die twee-tot-drie enkodeerder neem twee databisse en skakel dit om na 'n parallelle drie bis kode. Die uitset van die Trellis-enkodeerder volg 'n sekere patroon waarin die wins van die stelsel opgesluit lê. Daar bestaan verskillende maniere om die pad wat deur die trellis gevolg word voor te stel. Die eerste hiervan is die trellisdiagram.

3.2.4 Die trellisdiagram.

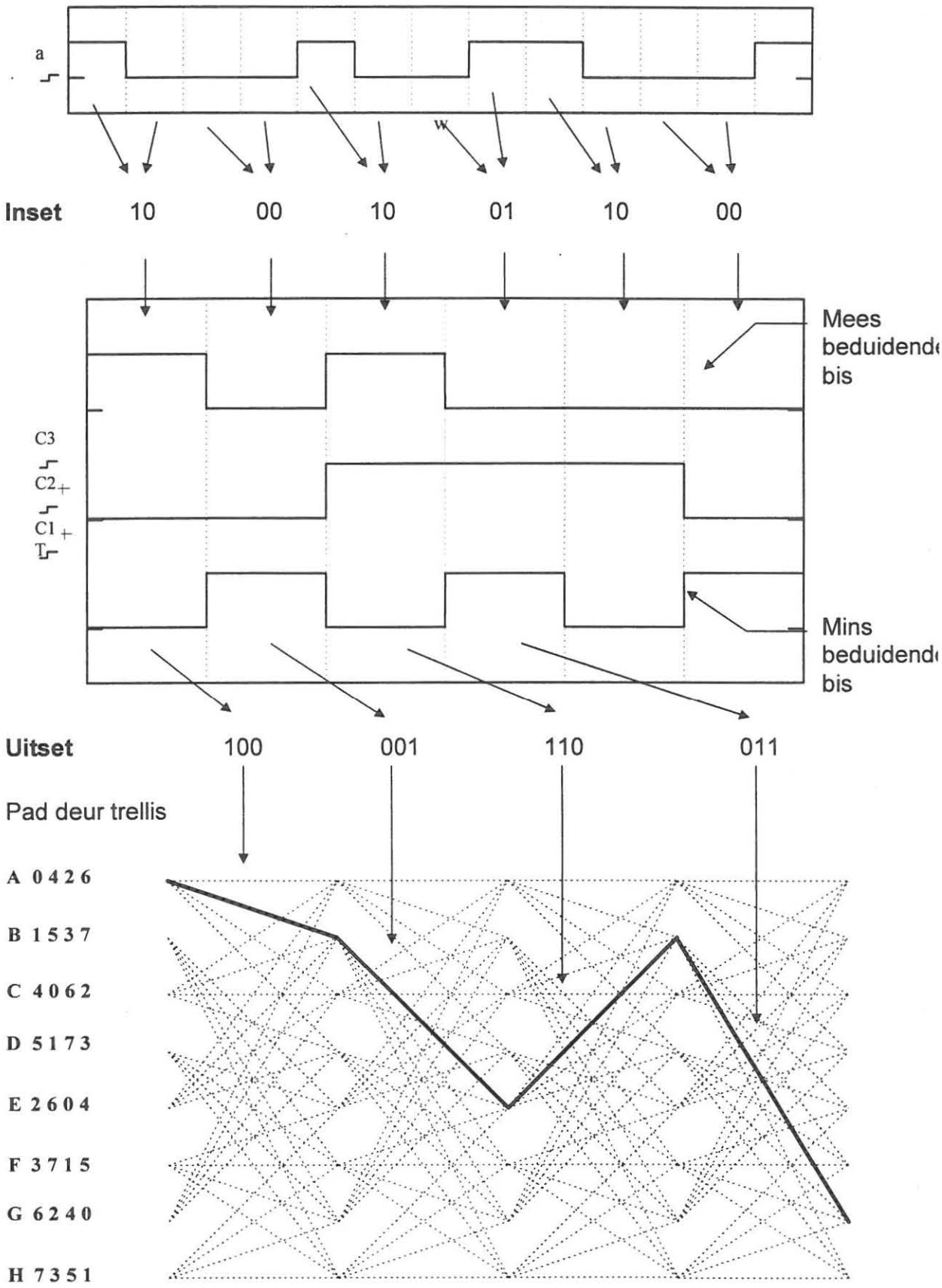
Ter verduideliking van die trellisdiagram is die data aan die bokant van Figuur 3-5 as inset van die enkodeerder in Figuur 3-4 gebruik. Die aanvanklike staat van die skuifregisters word as nul aanvaar. Die pad wat deur die trellis in Figuur 3-5 aangetoon word, is die uitset van die enkodeerder met 'n inset soos aangedui. Die soliede lyn dui die pad wat die data deur die trellis volg aan. Daar kan duidelik gesien word dat sekere spronge nie moontlik is nie. So kan van posisie A slegs na posisies A, B, C en D op die trellis beweeg word en nie na E, F, G of H nie. Dus word die aantal moontlike uitsetsimbole vir daardie spesifieke sprong beperk.

3.2.5 Die trelliskodeboom

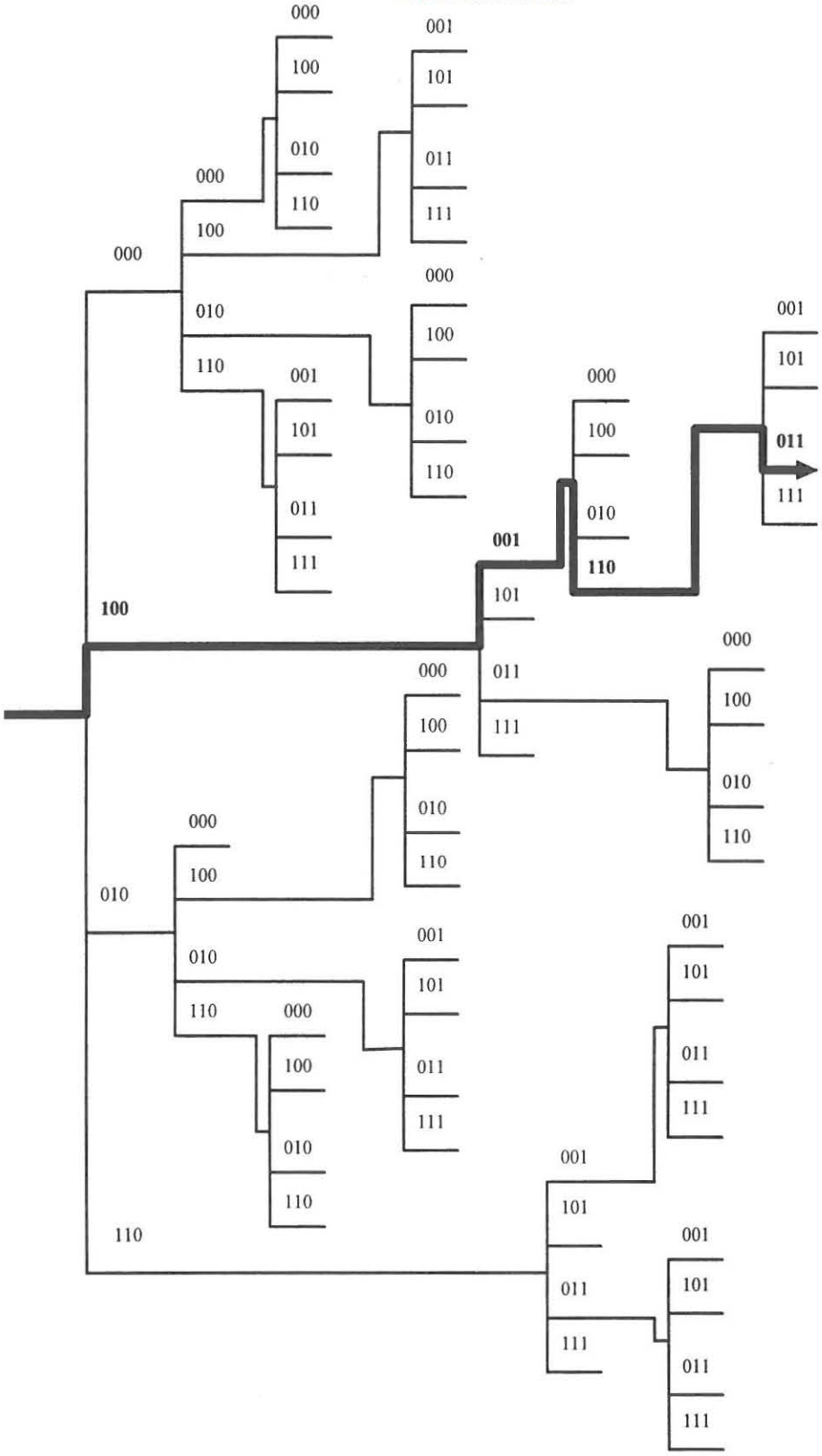
Die tweede moontlike voorstelling van die pad wat deur die trellis gevolg word is die trelliskodeboom soos voorgestel in Figuur 3-6. Ter verduideliking is die

[1] MathCAD is die eiendom van Mathsoft met kopiereg op die program.

uitsette van die enkodeerder in Figuur 3-5 gebruik om die pad deur die kodeboom aan te dui. Uit die voorbeeld kan daar egter waargeneem word dat dit bykans onmoontlik is om 'n redelike groot aantal stappe deur die kodeboom te volg. Dit kan toegeskryf word aan die feit dat die aantal vertakkings na elke stadium eksponensieel toeneem. Deur van die enkodeerder-uitset in Figuur 3-5 gebruik te maak kan die pad, soos wat op die kodeboom aangedui is, gevolg word. Die pad van die ge-enkodeerde data word met 'n beklemtoonde lyn aangedui.



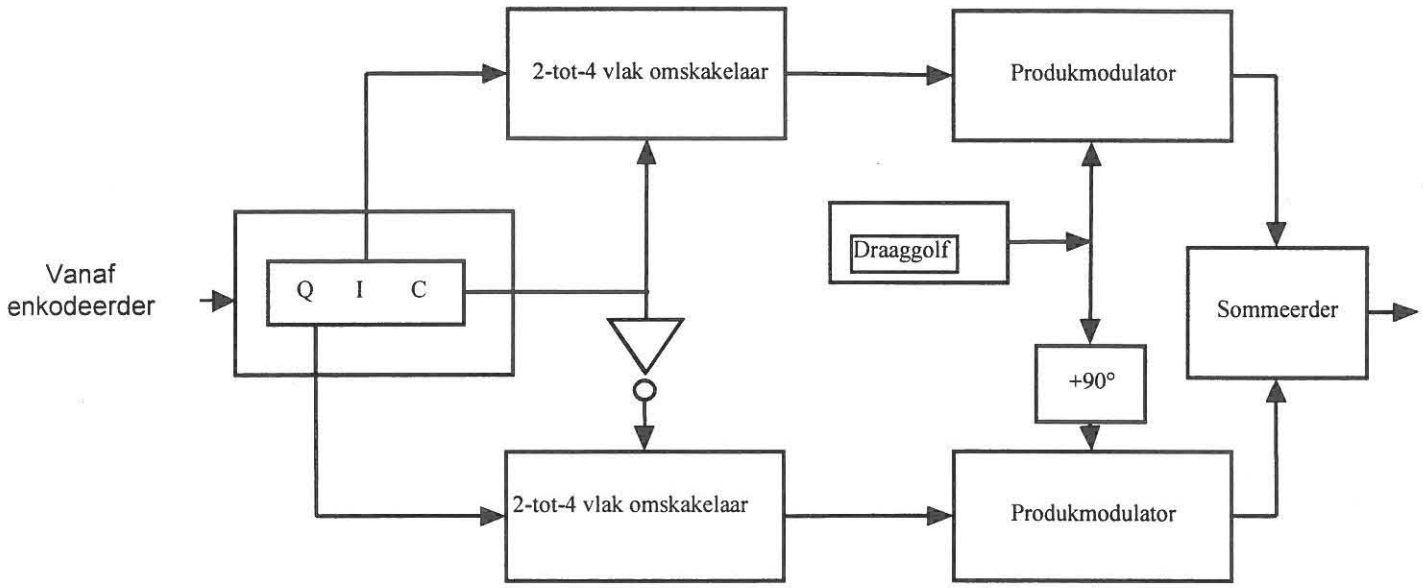
Figuur 3-5: Saamgestelde skets ter verduideliking van die trellis.



Figuur 3-6: Trelliskodeboom met tipiese insetdata. (Soos aangedui in Figuur 3-5)

3.2.6 Bespreking van 'n 8-PSS modulator.

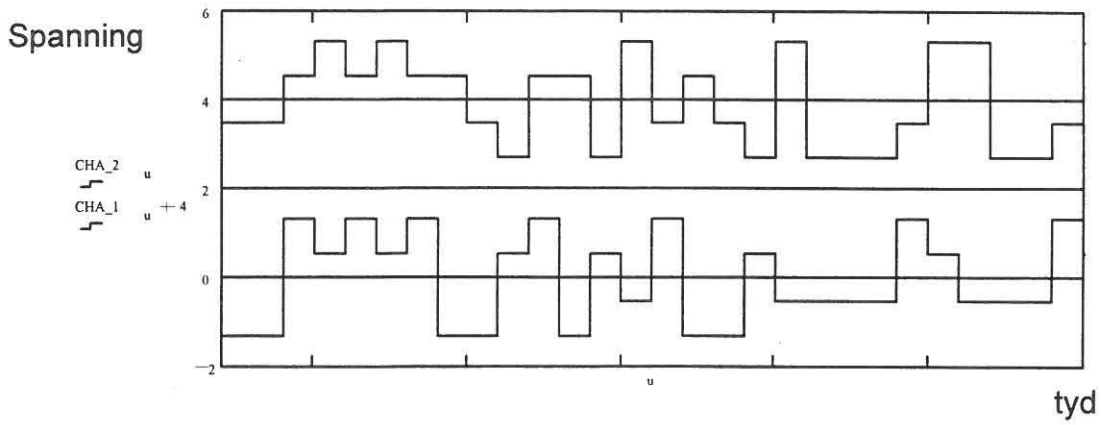
Die inset van die modulator word verkry vanaf die uitset van 'n 2-tot-3 enkodeerder. Die drie uitsetbisse van die enkodeerder, en dus die inset-elemente van hierdie kanaal, word na verwys as die I (infase) kanaal, Q (kwadratuur) kanaal en C (kontrole) kanaal waar Q die mees- en C die minsbeduidende uitset bis van die enkodeerder is.



Figuur 3-7: 8-PSS Modulator.

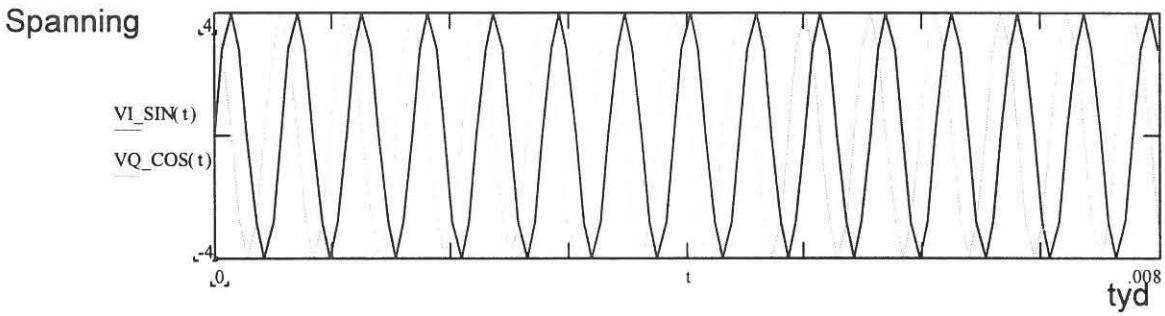
Hierdie I, Q en C is die insette van die 2-tot-4 vlak omskakelaars. Twee van die insetbisse word na elk van die twee-tot-viervlak omskakelaars se insette geneem. Die omskakelaars skakel dan die tweebis inset om na een van vier spanningsvlakke. Figuur 3-8 dui die uitset van die twee-tot-viervlak omskakelaars aan - soos volgens 'n MathCAD simulasie vir tipiese insetdata.

Q, I en C is onderskeidelik aan C_3, C_2 en C_1 van die enkodeerder, aangedui in Figuur 3-4, gekoppel



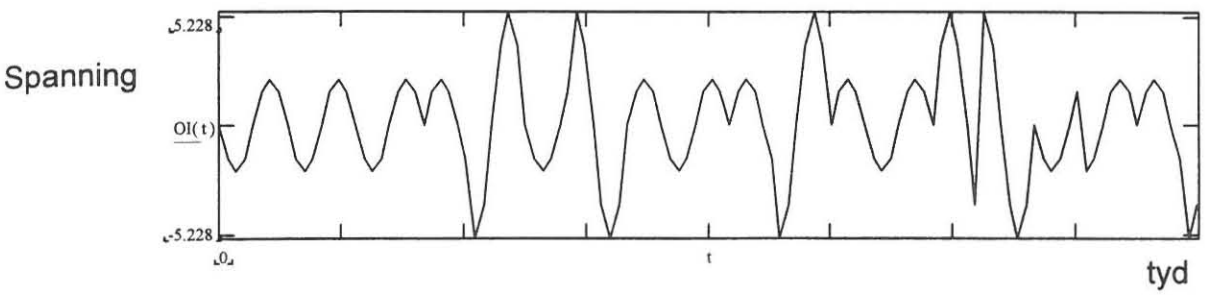
Figuur 3-8: Gesimuleerde uitset van die 2-tot-4-vlak omskakelaars met tipiese insette.

Na die 2-tot-4-vlak omskakelaars word die viervlak bisstrome aan die produkmodulators gekoppel. 'n Sinus en 'n cosinus sein word nou opgewek ten einde infase en kwadratuur draers te verkry. Die draagfrekwensie in die simulasie was 1800Hz. Die infase data word met sin(ωt) vermenigvuldig en die kwadratuur fase data word met sin(ωt + 90°).



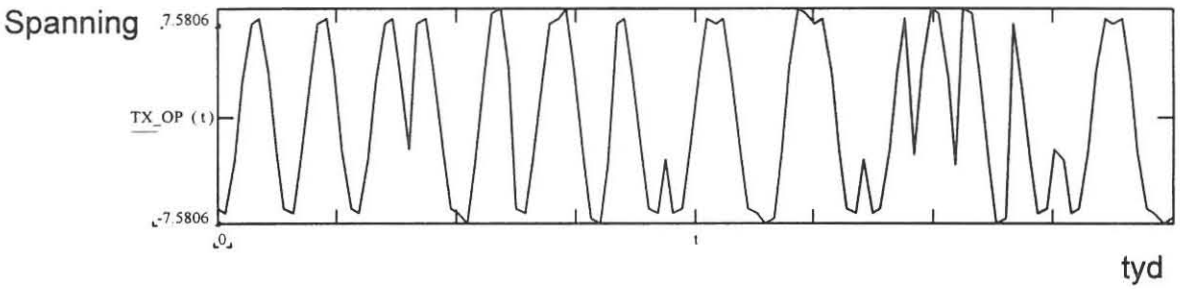
Figuur 3-9: Infase en kwadratuur draers.

Figuur 3-10 Toon die gesimuleerde uitset van een van die twee produkmodulators aan.



Figuur 3-10: Uitset van een van die produkmodulators.

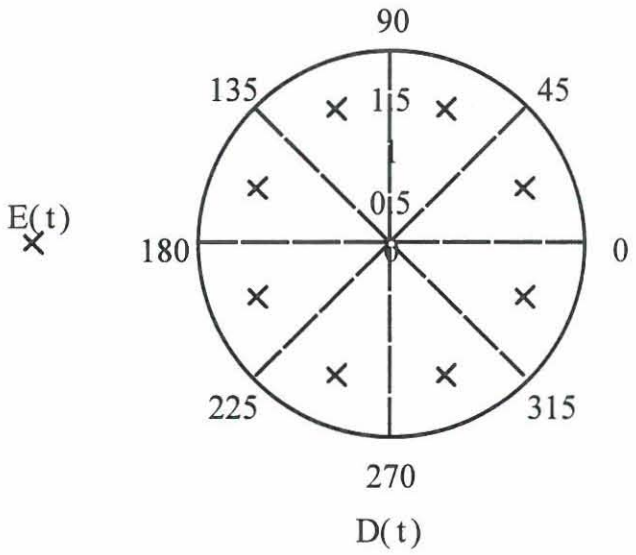
Ten einde die 8-PSS uitset te verkry is die uitsette van die twee produkmodulators gesommeer. Die gesommeerde uitset, soos in MathCAD gesimuleer kan in Figuur 3-11 gesien word. Die sein vertoon hoekig omdat die aantal monsters per siklus beperk was.



Figuur 3-11: Uitset van gesimuleerde 8-PSS sender.

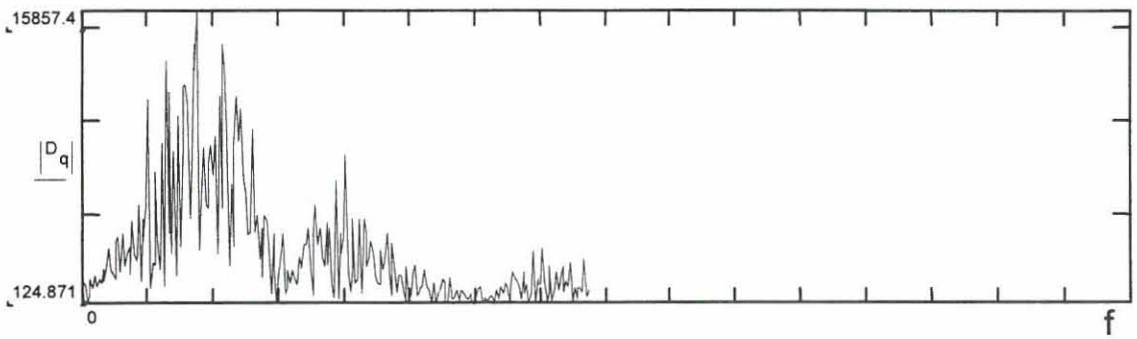
Daar kan duidelik op die uitset van die sommeerder waargeneem word dat slegs die fases van die opeenvolgende simbole verskil en 'n konstante amplitude geld. Hierdie gesommeerde uitset is die uitset van die 8-PSS sender.

Figuur 3-12 toon die uitsetfases van die 8-PSS sender aan.



Figuur 3-12: Die agt uitsetfases van die 8-PSS sender.

Die sein afkomstig van die sender moet egter eers filtreer word voor dit oor 'n kanaal met 'n beperkte bandwydte versend kan word. Figuur 3-13 dui die ongefiltreerde spektrum van die gemoduleerde sein aan.



Figuur 3-13: Frekwensiespektrum van die ongefilterde gemoduleerde 8-PSS sein.

Ten einde die korrekte filtrering te doen moet die benodigde bandwydte bereken word. Die benodigde bandwydte vir die 8-PSS sein kan dan as volg bereken word [**4**, p. 333].

$$B = \frac{2R_b}{\log_2 M} \tag{3-1}$$

Waar B = die bandwydte is.

M = die aantal moontlike fases van die PSS sein.

R_b = die bistempo.

Vereenvoudig kan die infase modulator uitset geskryf word as [20, p. 30].

$$\theta = (X \sin \omega_a t)(\sin \omega_c t)$$

Waar $\sin \omega_a t = 2\pi \frac{f_a t}{6}$ = die moduleersein is.

$\sin \omega_c t = 2\pi f_c t$ = die draagfrekwensie is.

Daarom is die gemoduleerde sein

$$\theta = \left(X \sin 2\pi \frac{f_b t}{6} \right)(\sin 2\pi f_c t)$$

$$= \frac{X}{2} \cos 2\pi \left(f_c - \frac{f_b}{6} \right) t - \frac{X}{2} \cos 2\pi \left(f_c + \frac{f_b}{6} \right) t$$

waar: f_b = Die bistempo.

f_c = Die draagfrekwensie.

X = Die amplitude van die sein op die modulator inset.

Die frekwensiespektrum strek dus van $\left(f_c - \frac{f_b}{6}\right)$ tot $\left(f_c + \frac{f_b}{6}\right)$.

Die minimum BW (F_n) is dus $f_n = \left(f_c + \frac{f_b}{6}\right) - \left(f_c - \frac{f_b}{6}\right) = \frac{2f_b}{6} = \frac{f_b}{3}$

Die eerste spektrale nul bo die draagfrekwensie kom dus voor by $\left(f_c + \frac{f_b}{3}\right)$. Die sein moet dus deur 'n laagdeurlaat filter met 'n afsnyfrekwensie van $\left(f_c + \frac{f_b}{3}\right)$ gefilter word. Die berekende BW van die betrokke stelsel, waar $f_c = 400Hz$ en $f_b = 1200$,sal dus van 0 tot 800Hz strek.

3.2.7 Demodulasie.

Die gefiltreerde uitset van die sender is die inset van die demodulator by die ontvanger. Sinchronisasie tussen die draers van die sender en die demodulator van die ontvanger is van die grootste belang. Hier word die ontvangde sein met die infase en die kwadratuurfase draers vermenigvuldig. Indien die fase van die verwysingsein, wat by die ontvanger opgewek word, se fase verskil van die oorspronklike draer sou dit nie moontlik wees om die ontvangde sein se fase korrek te bepaal nie.

Dieselfde frekwensie en fase wat vir modulasie van die sein gebruik is word benodig vir die suksesvolle demodulasie van die sein [7, p. 44].

Stremler beskryf die verskynsel as volg [18, p. 223]:

Neem 'n frekwensiefout as $\Delta\omega$ en 'n fasefout as $\theta\omega$ in die gegenereerde draer by die ontvanger. Die insetsein word geskryf as $a(t)\cos\omega_c t$

Die ontvanger genereer dan die volgende produk.

$$\phi(t) = a(t)\cos\omega_c t\cos[(\omega_c + \Delta\omega)t + \theta_o]$$

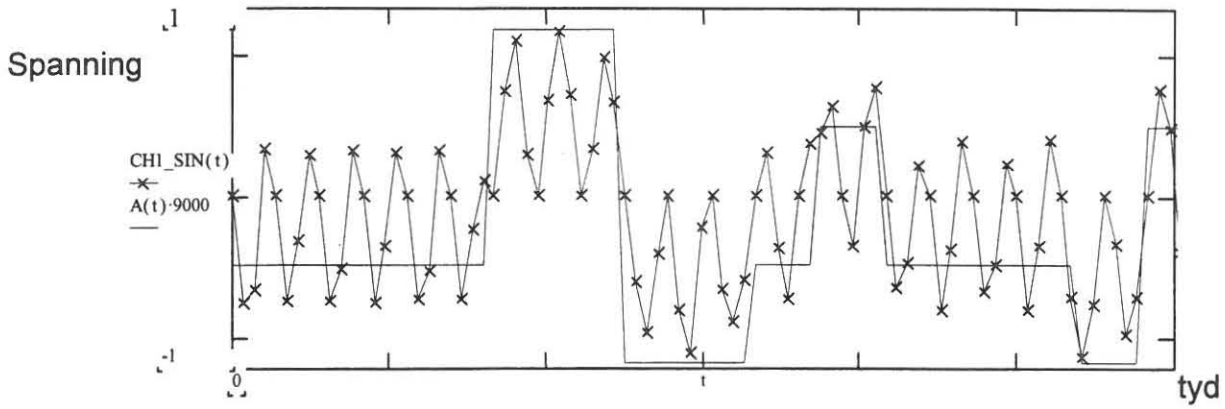
$$= \frac{1}{2}a(t)\cos[(\Delta\omega)t + \theta_o] + \frac{1}{2}a(t)\cos[(2\omega_c + \Delta\omega)t + \theta_o] \qquad (3\text{-}2)$$

Die regterhandse term is gesentreer by $\pm(2\omega_c + \Delta\omega)$ en word deur 'n laagdeurlaatfilter uitgefilter. Die uitset van hierdie filter lewer dan die volgende:

$$e_o(t) = \frac{1}{2}a(t)\cos[(\Delta\omega)t + \theta_o] \qquad (3\text{-}3)$$

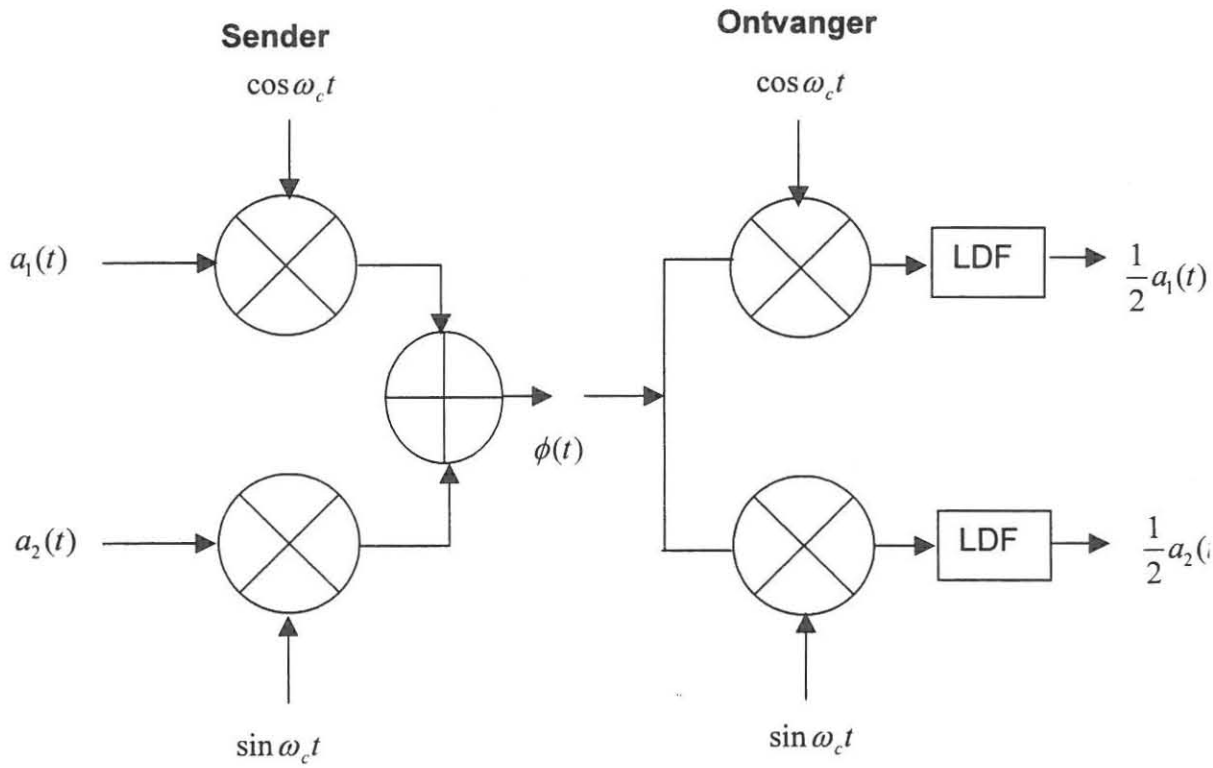
Die uitset is dus $\frac{1}{2}a(t)$, slegs as beide die fase en frekwensiefoute gelyk is aan 0. Uit bogenoemde is dit duidelik dat beide die fase en die frekwensie van die herwinde draer dieselfde moet wees as die van die sender.

Figuur 3-14 dui die ontvangde sein aan nadat dit met die infase draer vermenigvuldig is. Die oorspronklike data vlakke word ook aangedui.



Figuur 3-14: Voorstelling van gesimuleerde sein nadat met die infase draer vermenigvuldig is.

Ten einde die viervlak sein weer te herwin moet die hoë frekwensie komponent van die sein uitgefilter word. Die frekwensiekomponente teenwoordig in die stelsel word in Figuur 3-15 voorgestel.



Figuur 3-15 Voorstelling van seinposisies by die modulator en die demodulator in die 8-PSS stelsel.

Die uitset van die modulator, en dus die inset van die sender, is die volgende [18, p. 224].

$$\phi(t) = a_1(t)\cos\omega_c t + a_2(t)\sin\omega_c t \qquad (3\text{-}4)$$

Indien die sein in die demodulasieproses met $\cos\omega_c(t)$ vermenigvuldig word, word die volgende verkry:

$$\phi(t)\cos\omega_c t = a_1(t)\cos^2\omega_c t + a_2(t)\sin\omega_c t\cos\omega_c t \qquad (3\text{-}5)$$

$$= \frac{1}{2}a_1(t) + \frac{1}{2}a_1(t)\cos 2\omega_c t + \frac{1}{2}a_2(t)\sin 2\omega_c t \qquad (3\text{-}6)$$

Die uitset is dus die basisbandsein, plus twee komponente teen twee keer die draagfrekwensie. Indien die sein in die demodulasieproses met $\sin\omega\ (t)$ vermenigvuldig word word die volgende verkry:

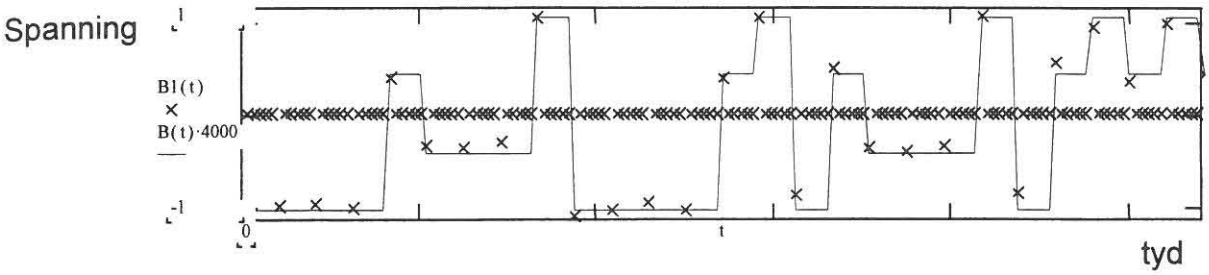
$$\phi(t)\sin\omega_c t = a_1(t)\cos\omega_c t\sin\omega_c t + a_2(t)\sin^2\omega_c t \qquad (3\text{-}7)$$

$$= \frac{1}{2}a_1(t)\sin 2\omega_c t + \frac{1}{2}a_2(t) - \frac{1}{2}a_2(t)\cos 2\omega_c t \qquad (3\text{-}8)$$

Indien 'n laagdeurlaatfilter gebruik word sal alle terme wat $2\omega_c$ bevat verswak word. Dus sal $\frac{1}{2}a_1(t)$ en $\frac{1}{2}a_2(t)$ oorbly wat gelyk is aan die moduleersein teen die helfte van die amplitude.

'n Manier om die laagdeurlaatfilter te implementeer is om die gemiddeld van die monsters oor 'n sekere periode te bepaal, hierdie periode moet egter

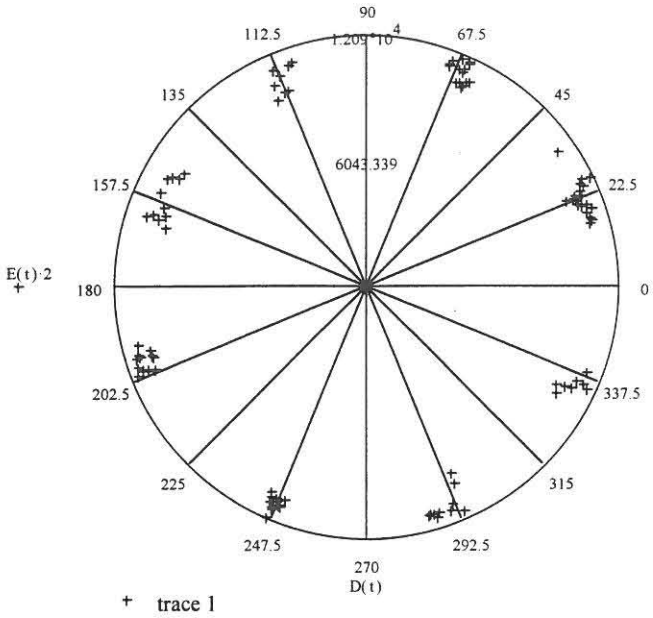
gesinchroniseerd wees met die simboolperiode. Dit is ook 'n metode wat in DSP geïmplementeer kan word. Figuur 3-16 dui die gemiddeld van agt opeenvolgende monsters op die eerste monsterposisie aan (let daarop dat die res van die monsters nul gemaak is).



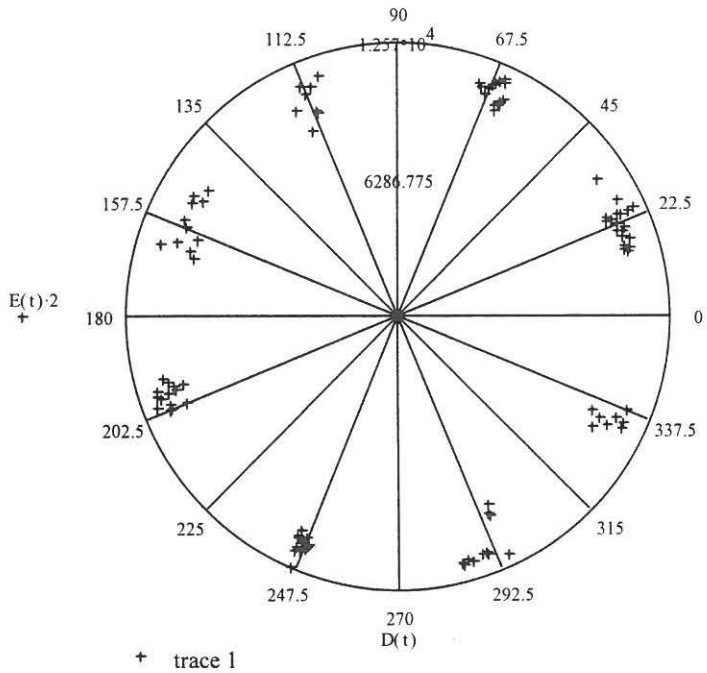
Figuur 3-16: Die gemiddeld van die agt monsters word op die eerste monsterposisie aangedui.

Die invloed van ruis, sowel as die beperking ten opsigte van bandwydte, kon ook duidelik deur middel van die simulasie op die fasordiagram gemonitor word. Die draagfrekwensie in die simulasies was 1800Hz. Figuur 3-17 tot Figuur 3-20 dui die gesimuleerde relatiewe posisie van die sein op die konstellasiediagram vir 'n kort datastroom aan.

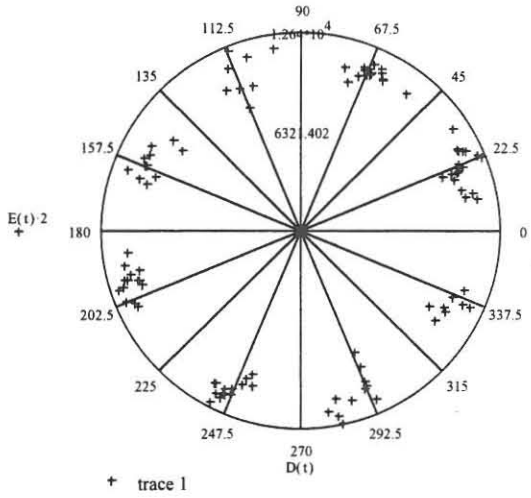
Die invloed van ruis teen verskillende seinruisverhoudings is ondersoek en die resultate daarvan word deur die volgende figure getoon.



Figuur 3-17: Die gesimuleerde ontvangde 8-PSS sein indien dit deur 'n filter met 'n bandwydte van 3400Hz gefiltreer is en daar geen ruis bygevoeg is nie.

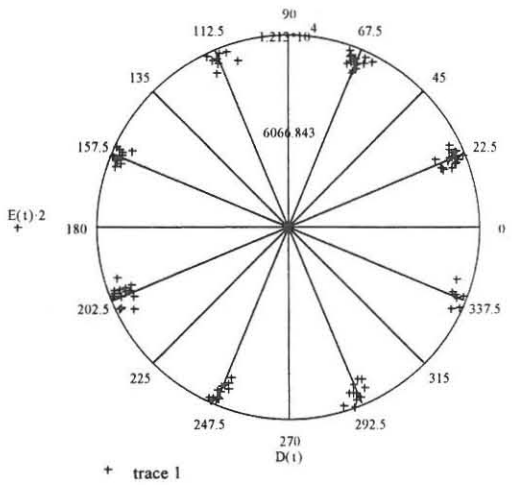


Figuur 3-18: Die gesimuleerde ontvangde 8-PSS sein indien dit deur 'n filter met 'n bandwydte van 3400Hz gefiltreer is teen 'n seinruisverhouding van 10dB.



Figuur 3-19: Die gesimuleerde ontvangde 8-PSS sein indien dit deur 'n filter met 'n bandwydte van 3400Hz gefiltreer is teen 'n seinruisverhouding van 6.9dB.

'n Duidelike verswakking in die verspreiding van ontvangde simbole kan waargeneem word met 'n toename in ruis. Ten einde die effek van bandwydte op die stelsel te bestudeer is die seinruisverhouding op 6.9dB gehou en die bandwydte vergroot. Daar kan duidelik in Figuur 3-20, relatief tot Figuur 3-19, waargeneem word dat 'n vergroting in bandwydte 'n verbetering in die groepering van ontvangde fases veroorsaak.



Figuur 3-20: Die gesimuleerde ontvangde 8-PSS sein indien dit deur 'n filter met 'n bandwydte van 7200Hz gefiltreer is met 'n seinruisverhouding van 6.9dB.

3.3 Viterbi dekodering

Viterbi dekodering het een van die mees algemene dekoderingstegnieke geword. Dit kan grotendeels toegeskryf word aan die gemak van implementering en die wins wat uit die stelsel verkry word [9, p. 698]. Indien verskillende dekoderingstegnieke teen mekaar opgeweeg word blyk dit dat die Viterbi mees-waarskynlikheidstegniek die beste presteer. Hierdie stelsel is veral waardevol waar effektiewe energieverbruik belangrik is [9, p. 698].

By dekodering van konvolusiekodes is die probleem om 'n pad deur die trellisdiagram[2] te vind deur gebruik te maak van sekere dekoderingsreëls. Die ideaal is om 'n pad te kies wat, indien die uitset van die enkodeerder met dié van die dekodeerder vergelyk word, die minimum aantal simboolfoute sal voorkom. Omrede die aantal moontlike paaie deur die trellis eksponensieel met die aantal bisse toeneem, blyk dit moeilik te wees om die regte pad deur die trellis te vind. Wat egter deur die Viterbi-stelsel gedoen word is om sekere paaie te elimineer en na elke stadium sekere wenpaaie te kies. Op hierdie manier is daar na elke stadium slegs 'n sekere aantal wenners en vermeerder dit nie na elke stap nie.

Aan die einde van 'n bepaalde periode word die pad met die kortste afstand gekies en geneem as die waarskynlikste. Daar word dan teruggespoor deur die matriks en die nodus aan die begin van die wennerpad word as die wennernodus geïdentifiseer.

[2] Sien Figuur 3-5 ter verduideliking van trellisdiagram.

Indien 'n trellisge-enkodeerde 8-PSS sein deur 'n Viterbi-ontvanger ontvang moet word, is die eerste stap om minimum afstande tussen die ontvangde fase en die moontlike fases[3] te bepaal. Hierdie afstande kan deur middel van harde of sagte besluitneming bepaal word. Vervolgens word harde en sagte besluitneming bespreek.

3.3.1 Viterbi-dekodering (Harde besluitneming).

Indien van harde besluitneming gebruik gemaak word, neem die demodulator die inkomende fase en bepaal watter een van die agt fases die naaste aan die ontvangde fase is. Hierdie fase word dan gesien as die ontvangde fase. Vervolgens word die Hammingafstand tussen die ontvangde fase en al die teoreties moontlike ontvangde fases bepaal. Hierdie Hammingafstande word dan in 'n matriks gestoor. Na 'n bepaalde aantal ontvangde fases word daar teruggespoor deur die matriks en die pad met die kortste moontlike totale Hammingafstand word gekies as die wennerpad en word as die mees waarskynlike blok data aanvaar [14, p. 191].

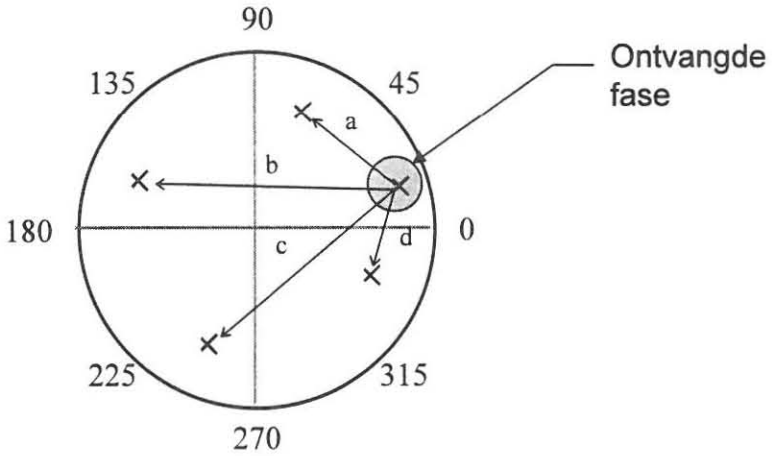
Data wat op hierdie manier getoets en ontfout word se moontlikheid op verdere foute is skraal [4, p. 403]. Die grootte van die blok data wat vergelyk word om die minimum Hamming-afstand te bepaal, bepaal ook die akkuraatheid van die stelsel. Hoe groter hierdie blok is, hoe meer ingewikkeld en lomp raak die stelsel [12, p. 454]. Die basiese beginsel van die stelsel is om die ontvangde datapad te vergelyk met al die moontlike versende paaie en dan die datapad te kies wat die kortste Hammingafstand het [14, p. 191].

[3] Sien Figuur 3-1 vir moontlike fases.

3.3.2 Viterbi-dekodering (Sagte besluitneming).

Deur gebruik te maak van sagte besluitneming word in sommige gevalle 'n wins van +-2 dB verkry bo die van harde besluitneming [17, p. 354]. By die gebruik van sagte besluitneming word die euklidiese afstand tussen die ontvangde simbool en die moontlike ontvangde simbole bepaal.

Ter verduideliking van die euklidiese afstand word aanvaar dat die fase soos aangetoon in Figuur 3-21, deur die ontvanger ontvang word. Die trellis enkodeerder het die gevolg gehad dat slegs die vier fases soos in die figuur aangetoon, ontvang kon word.



Figuur 3-21: Bepaling van euklidiese afstand.

Die werklike afstand tussen die ontvangde en die moontlike fases word bepaal, naamlik die afstande van vektore a, b, c en d. Hierdie afstande word die euklidiese afstande genoem en word dan in 'n matriks gestoor. Wanneer by die terugspoorpunt gekom word, word die pad met die kortste totale afstand gekies.

Die terugsporing geskied nou op hierdie pad en die nodus aan die begin van die pad word as die wennernodus geïdentifiseer.

3.4 Opsomming

Indien die kanaalkapasiteit nie oorskry word nie is dit moontlik om deur die byvoeging van kodes deur die enkodeerder, dit vir die dekodeerder moontlik te maak om die data tot 'n groter mate van akuraatheid te herwin [21, p. 5]. In hierdie hoofstuk is die invloed van ruis sowel as bandwydte op die werking van 'n 8-PSS stelsel ondersoek en gesimuleer.

4 Uiteensetting van hardeware.

Vervolgens word die hardeware wat in die projek gebruik is beskryf.

4.1 Eksperimentele opstelling.

Die eksperimentele opstelling van die sender, die ontvanger en die kanaal is soos uiteengesit in Figuur 4-1.

- Beide die sender en die ontvanger is in een gasheerrekenaar monteer. 'n SIG-56 DSP kaart is in beide gevalle gebruik.

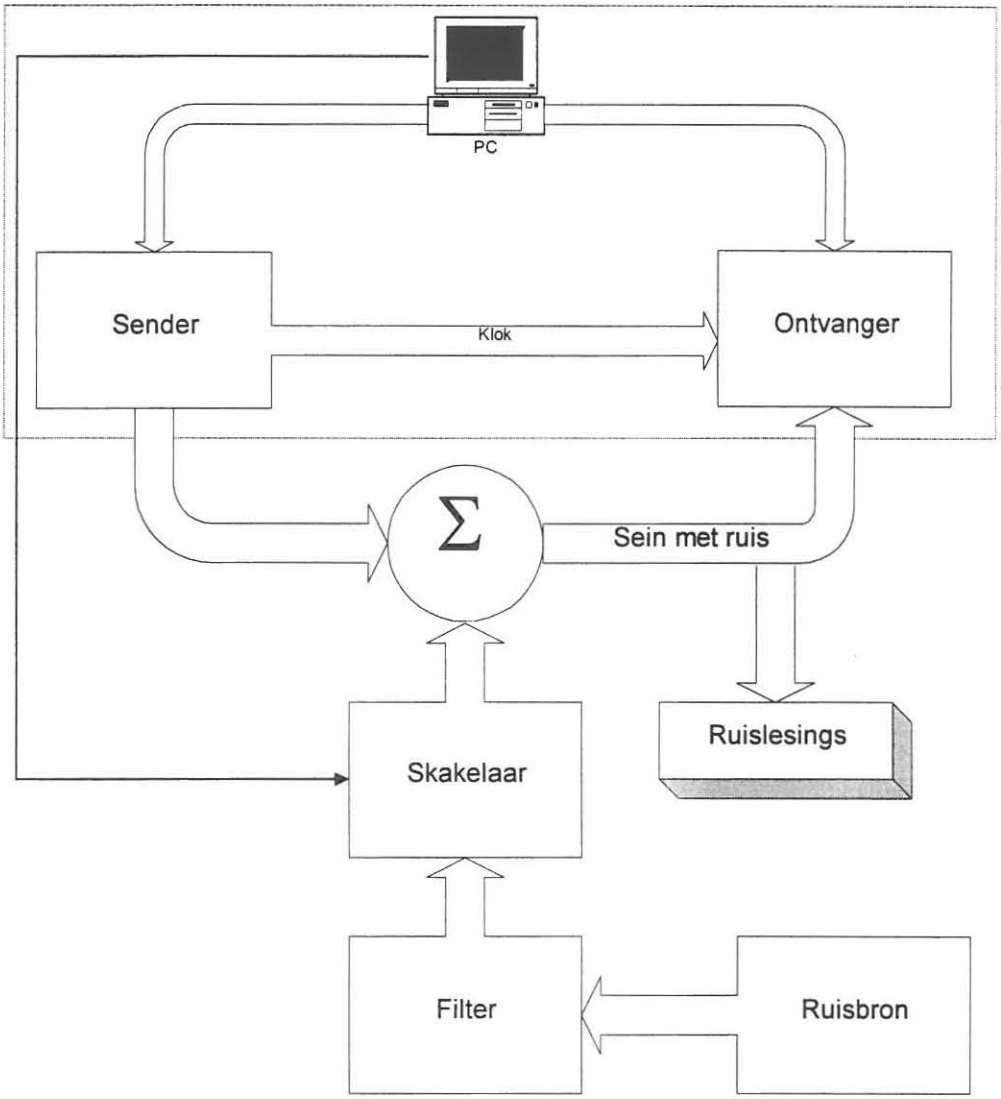
- Aangesien die projek nie gehandel het oor die sinchronisering van die stelsel nie, is die sender se klok deur middel van 'n eksterne verbinding aan die ontvanger gekoppel. Een ossillator het dus altwee die kaarte aangedryf.

- 'n Eksterne ruisbron is deur 'n elektroniese skakelaar aan die transmissielyn gekoppel. Omdat die sender en die ontvanger eers moes sinchroniseer, het hierdie skakelaar[4] eers ruis by die versende sein gevoeg nadat sinchronisasie plaasgevind het. Die ruisbron se uitset word deur 'n laagdeurlaatfilter gefiltreer voor dit by die sein gevoeg word. Dit verseker dat ruis met dieselfde bandwydte as die 8-PSS sein, daarby gevoeg word.

- Die sein en die ruis word bymekaar gevoeg deur 'n sommeerder.

- Hierdie saamgestelde sein word nou aan die ontvanger gekoppel.

- Die sein en ruis lesings word by die uitset van die sommeerder geneem.



Figuur 4-1: Eksperimentele opstelling.

[4] Die werking van die skakelaar word in paragraaf **6.3** meer breedvoerig behandel

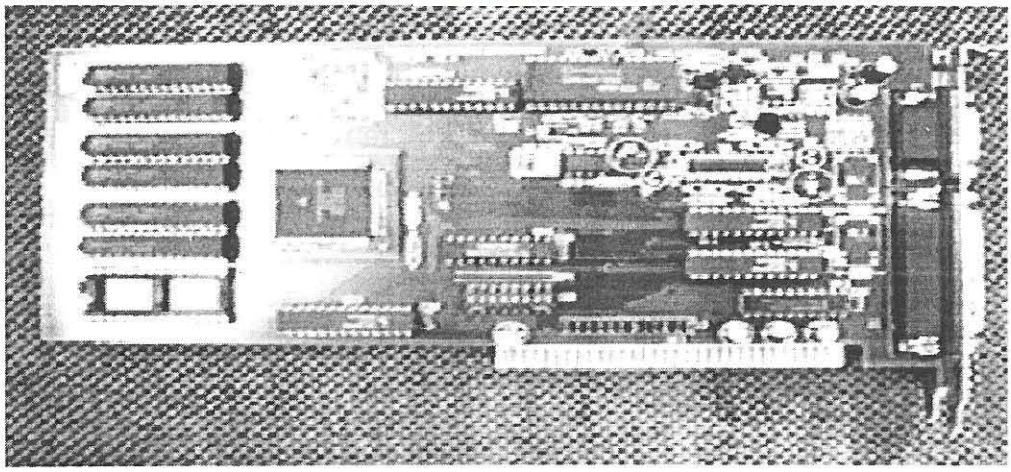
4.2 Die gasheerrekenaar

Die spesifikasies van die gasheerrekenaar waarin die sender sowel as die ontvanger DSP-kaarte gehuisves was, is as volg.

Prosesseerder:	486 DX 100
Ram:	16 Megagreep
Skerm:	VGA

4.3 SIG-56-kaarte as sender en ontvanger.

Vir die implementering van die sender sowel as die ontvanger is daar van SIG-56 ontwikkelingskaarte - wat deur PERALEX Electronic Development vervaardig is - gebruik gemaak. Die kaarte is gebaseer op die DSP56001 geïntegreerde stroombaan. Beheersagteware is in Pascal geskryf en beheer die oordrag van saamsteltaalprogramme vanaf die rekenaar na die SIG-56 kaarte. Die oordrag van data tussen die rekenaar en die kaarte word ook deur hierdie beheersagteware beheer. Figuur 4-2 is 'n foto van die SIG-56 DSP ontwikkelingskaart .



Figuur 4-2: SIG-56 DSP kaart soos aangewend as sender en ontvanger.

4.3.1 DSP56001 geïntegreerde stroombaan.

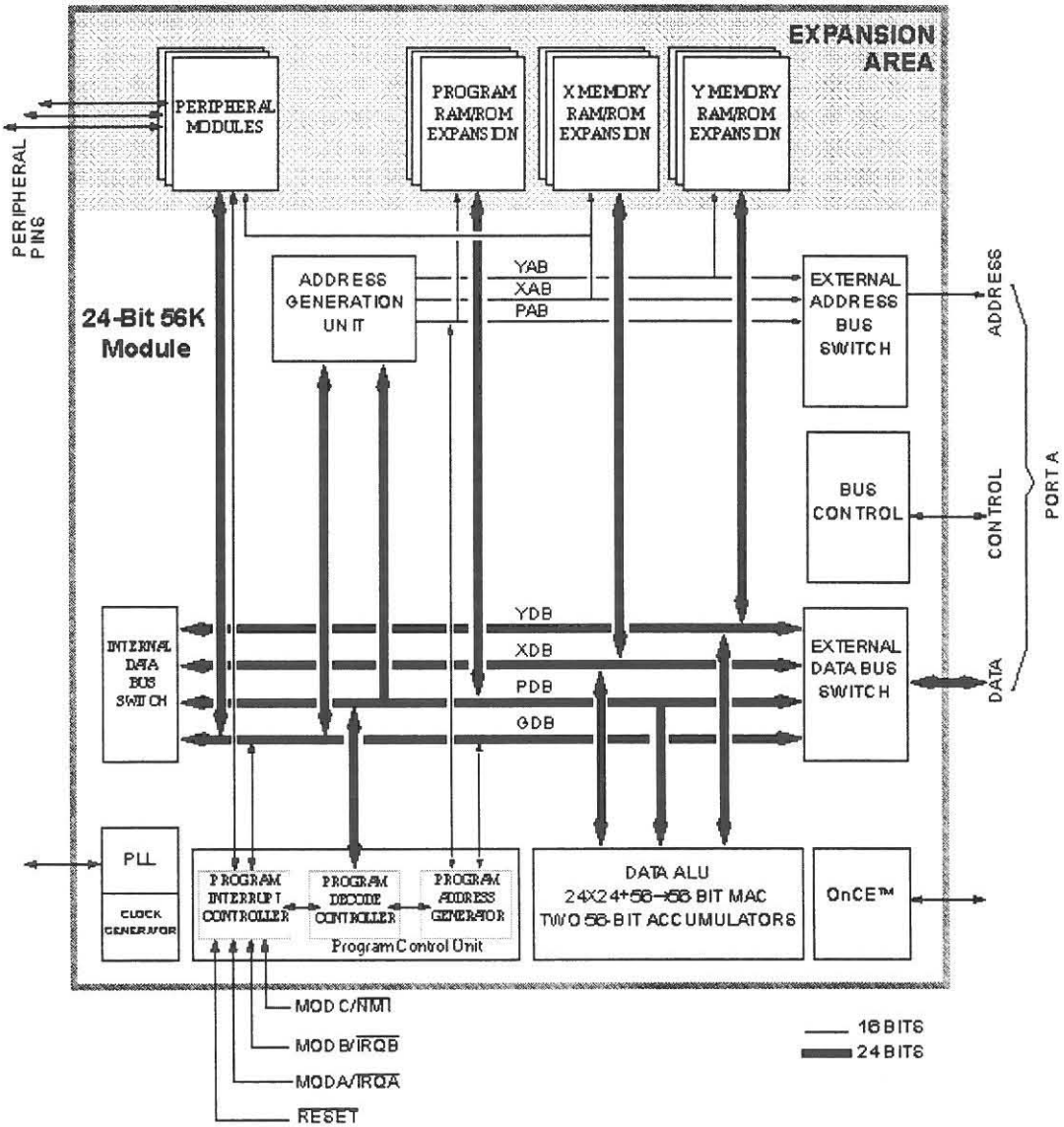
Vervolgens word 'n oorsig gegee van die uitleg van die DSP56001 geïntegreerde stroombaan. Die inligting is verkry vanaf 'n MOTOROLA internet datablad [10, p. 4].

4.3.1.1 Die databusse

Dataoordrag tussen die verwerkingseenhede geskied deur middel van vier 24-bis databusse. Hierdie databusse is:

- Die X-databus (XDB).
- Die Y-databus (YDB).
- Die program-databus (PDB).
- Die globale-databus (GDB).

Sekere instruksies gebruik die X en Y-databusse as een 48-bis databus. Data-oordrag tussen die rekenkundige en logiese eenheid en die X of Y-data geheue geskied deur YDB en XDB. Ander data-oordragte, soos byvoorbeeld na die poorte, geskied deur die GDB. Die busstruktuur ondersteun register-na-register, register-na-geheue en geheue-na-register oordragte.



Figuur 4-3: Blokdiagrammatiese voorstelling van die interne uitleg van die DSP56001 [10, p. 4].

4.3.1.2 Adresbusse

Adresse word gespesifiseer vir 'n interne X-datageheue en 'n interne Y-datageheue. Dit geskied op twee 16-bis busse, naamlik die X-adresbus en die Y-adresbus. Programgeheue word aangedui deur 'n bidireksionele program-adresbus.

4.3.1.3 Rekenkundige en logiese eenheid (RLE)

Al die wiskundige en logiese bewerkings word in die RLE gedoen. Die RLE bestaan uit vier 24-bis insetregisters, twee 48-bis akkumulatorregisters, twee 8-bis akkumulator-vergrotingsregisters en 'n vermenigvuldigingsakkumulator.

4.3.1.4 Die poorte

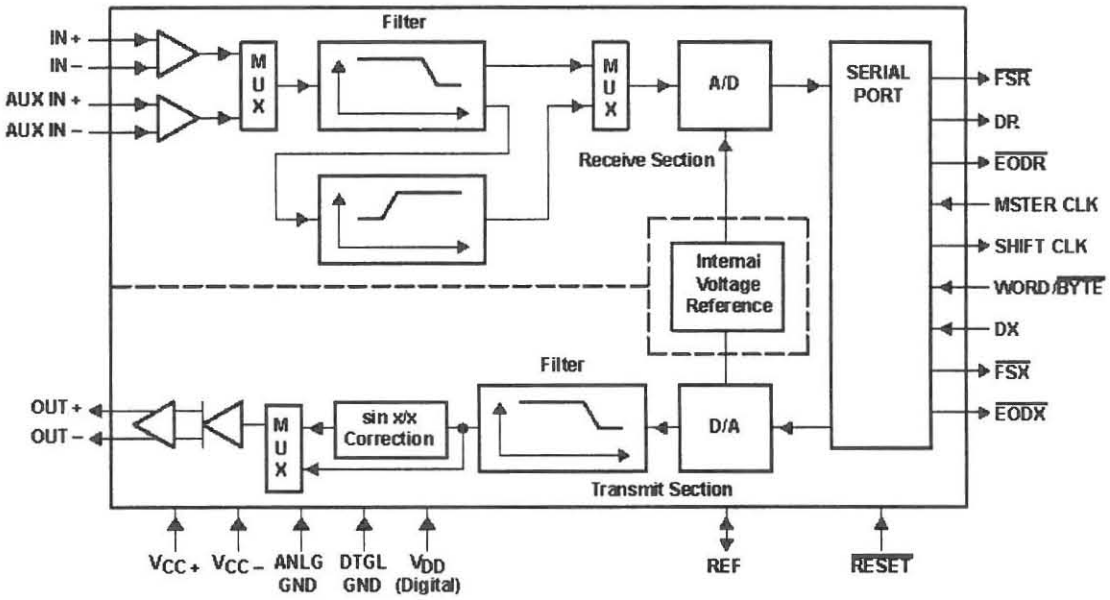
- **Poort A (Geheue-uitbreidingspoort)** kan koppel met 'n wye reeks koppelvlak-eenhede deur 'n 24-bis databus. Hierdie eenhede sluit hoëspoed statiese lees/skryf geheues, stadiger geheue eenhede en ander DSP eenhede in. Die uitbreidingsbus se klok is programmeerbaar en kan dus aangepas word om by verskillende stelsels aan te pas.

- **Poort B** is die gasheer koppelvlak en is 'n 8-bis voldupleks poort vir kommunikasie met die gasheer rekenaar.

- **Poort C** is 'n koppelvlak vir seriale kommunikasie (SCI) en sinkrone seriale kommunikasie (SSI). Die SCI is voldupleks vir 8-bis seriale kommunikasie met ander modems. Die SSI is meer aanpasbaar en kan deur die gebruiker verander word vir kommunikasie met 'n verskeidenheid ander toestelle.

4.3.2 TLC32044 analoog-na-digitaal en digitaal-na-analoog omskakelaar.

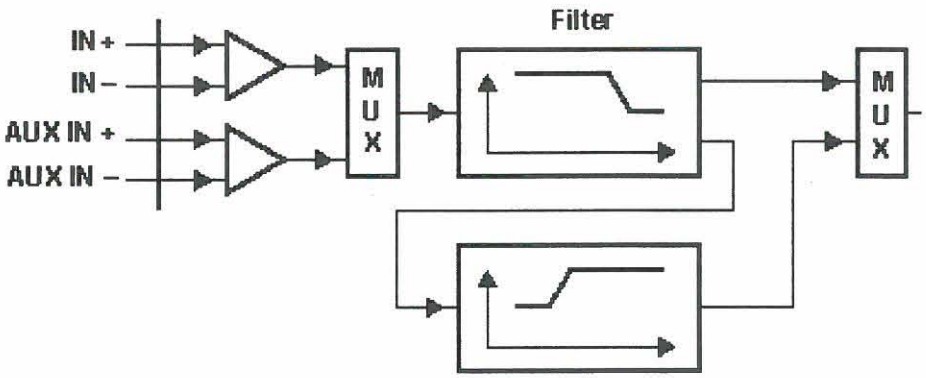
Die beskrywing van die TLC32044 is gebaseer op 'n datablad vanaf TEXAS INSTRUMENTS [19]. Die TLC32044 is 'n volledige analoog-na-digitaal en digitaal-na-analoog omskakelaar. Die koppeling tussen die TLC32044 en die DSP56001 geskied deur middel van 'n seriale poort. Daar is vier modusse waarin die seriale kommunikasie kan geskied. Die resolusie van die D-na-A en A-na-D is beide 14 bisse. Die TLC32044 het 'n programmeerbare wins en kan monster teen 19200 monsters/sekonde [19, p. 1]. Figuur 4-4 toon die interne funksionele blokdiagram van die TLC32044.



Figuur 4-4: Die blokdiagram van die TLC32044 [19, p.3].

4.3.2.1 Insetfilter

Die insetfilters bestaan onderskeidelik uit 'n agste en vierde orde Chebyshev hoog- en laagdeurlaatfilter. Die filters is in skakelkapasitortegnologie geïmplementeer en is programmeerbaar. Wanneer slegs 'n laagdeurlaatfilter verlang word kan die hoogdeurlaatfilter uitgeskakel word. Daar is twee analoog insette vir die analoog-na-digitale omskakelaar. Hierdie insette kan afsonderlik deur middel van sagteware geselekteer word. Figuur 4-5 [19, p. 17] dui blokdiagrammaties die koppeling aan.



Figuur 4-5: Die insetfilterkonfigurasie van die TLC32044 [19, p. 17].

Die boonste afsnyfrekwensie van die filter word as volg bereken.

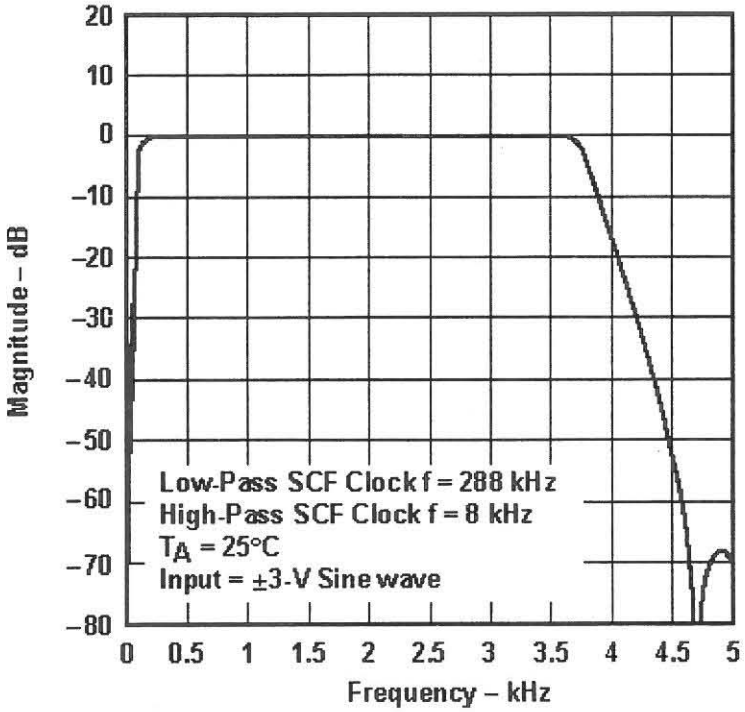
$$Afsnyfrekwensie = f_N \times \frac{f_{SCF}}{288kHz} \qquad (4\text{-}1)$$

Waar:

f_N = Genormaliseerde afsnyfrekwensie.

f_{SCF} = Skakelkapasitor klokfrekwensie.

Figuur 4-6 [19, p.29] dui die frekwensieweergawe van die banddeurlaarfilter aan indien die skakelkapasitorfilter se klokfrekwensie 288kHz, en die monsterfrekwensie 8kHz is.



Figuur 4-6: Frekwensieweergawe van banddeurlaatfilter [19, p.34].

Die TLC32044 maak voorsiening vir verskillende monsterfrekwensies. Figuur 4-7 dui die interne tydkonfigurasie aan [19, p.9]. Die skakelkapasitorfrekwensie word as volg bereken [19, p.10].

$$f_{SCF} = \frac{f_M}{2 \times (tellerA)} \qquad (4\text{-}2)$$

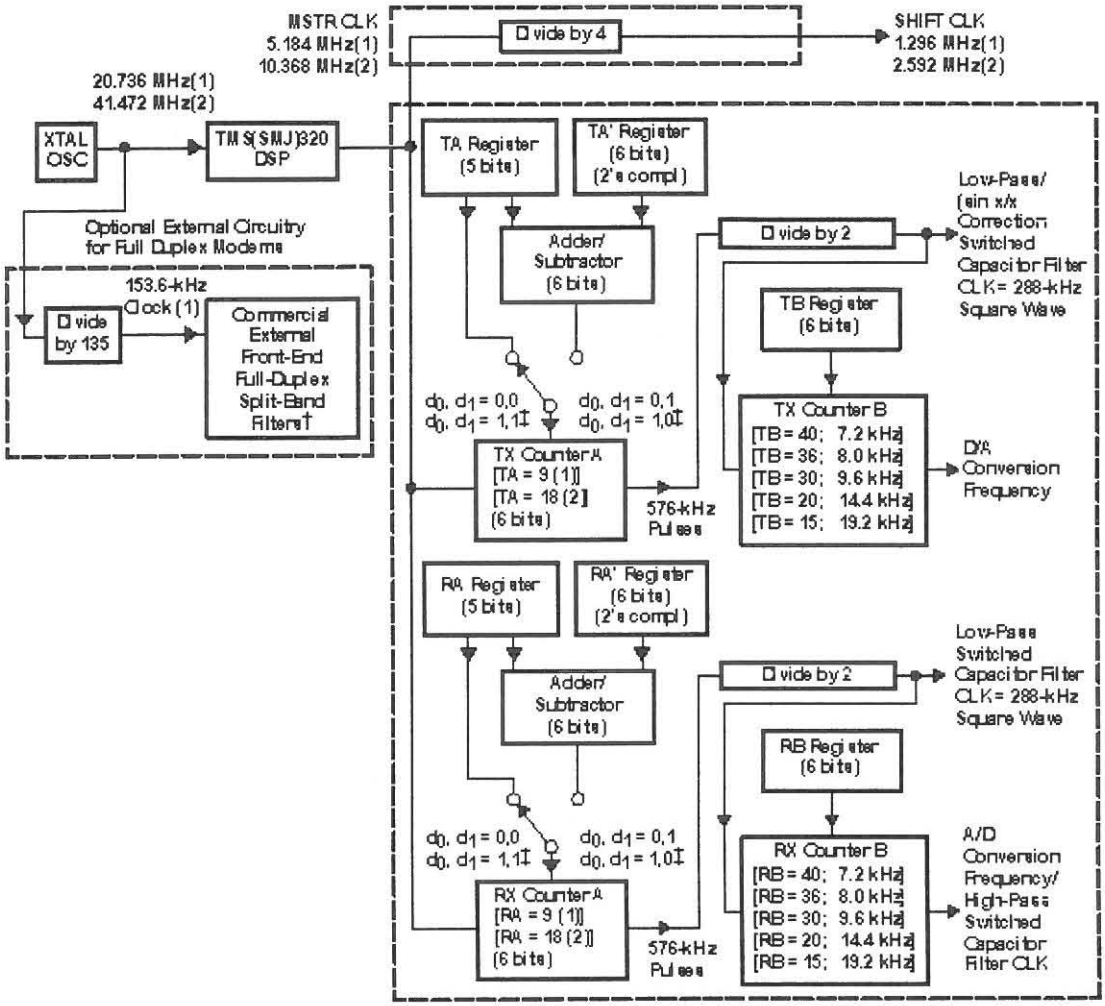
Waar: f_M = Meesterklokfrekwensie.

Die monsterfrekwensie word as volg bereken.

$$f_s = \frac{f_{SCF}}{(tellerB)} \qquad (4\text{-}3)$$

Waar:

f_S = Monsterfrekwensie.

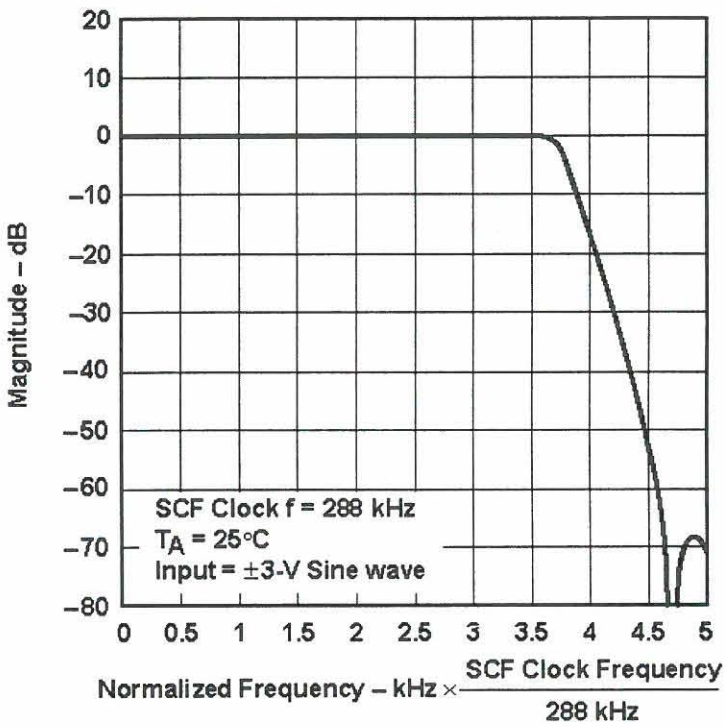


Figuur 4-7: Interne tyddiagram van die TLC32044 [19, p.9].

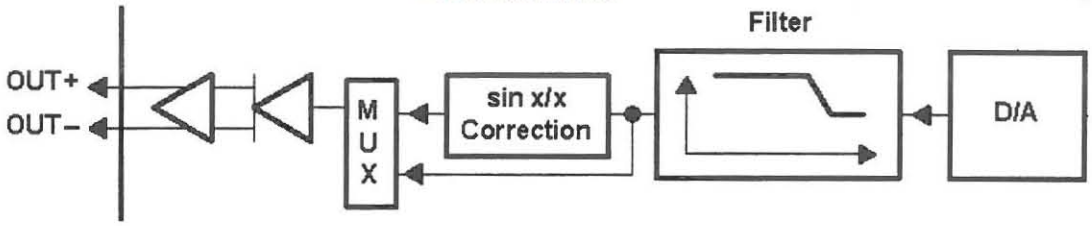
4.3.2.2 Uitsetfilter

Op die uitset van die digitaal-na-analoog omskakelaar is daar 'n agtste-orde Chehychev laagdeurlaatfilter. Figuur 4-8 [19, p.29] dui die frekwensieweergawe van die uitsetfilter aan vir 'n skakelkapasitorfrekwensie van 288kHz.

Die boonste afsnyfrekwensie, sowel as die omskakelingstempo van die D-na-A omskakelaar, word op dieselfde wyse bepaal as by die insetfilter (sien paragraaf 4.3.2.1). Die enigste verskil is dat sendregisters (TA) en (TB) gebruik word. Figuur 4-9 [19, p. 3] dui die blokdiagram van die uitsetkonfigurasie van die TLC32044 aan.



Figuur 4-8: Frekwensieweergawe van die laagdeurlaatfilter op uitset van D-na-A omskakelaar [19, p. 34].



Figuur 4-9: Blokdiagram van uitsetfilterkonfigurasie [19, p. 3].

Op die uitset van die filter is daar 'n (sin x)/x korreksiefilter wat steurings uitskakel wat op veelvoude van die skakelkapasitorfrekwensie voorkom [6, p. 289]. Die (sin x)/x filter kan deur middel van sagteware aan of af geskakel word.

5 Sagteware

5.1 Beheersagteware

Die beheersagteware beheer die sender en ontvangerkaarte en is in Pascal geskryf. Dit verrig die volgende funksies.

- Die sender se sagteware word deur die beheersagteware vanaf die gasheerrekenaar na die DSP (sender) kaart oorgedra.
- Die ontvanger se sagteware word vanaf die gasheerrekenaar na die DSP (ontvanger) kaart oorgedra.
- Ewekansige datawoorde word deur die beheersagteware gegenereer en aan die senderkaart oorgedra.
- Ontvangde datawoorde word vanaf die ontvangerkaart na die gasheerrekenaar oorgedra.
- Ontvangde data word met versende data vergelyk ten einde die aantal bisfoute te bepaal.

5.2 Sender sagteware

Hier volg 'n beskrywing van die sender sagteware wat in saamsteltaal geskryf is. Die sagteware word aan die hand van die vloeidiagram in Figuur 5-1 bespreek.

5 Sagteware

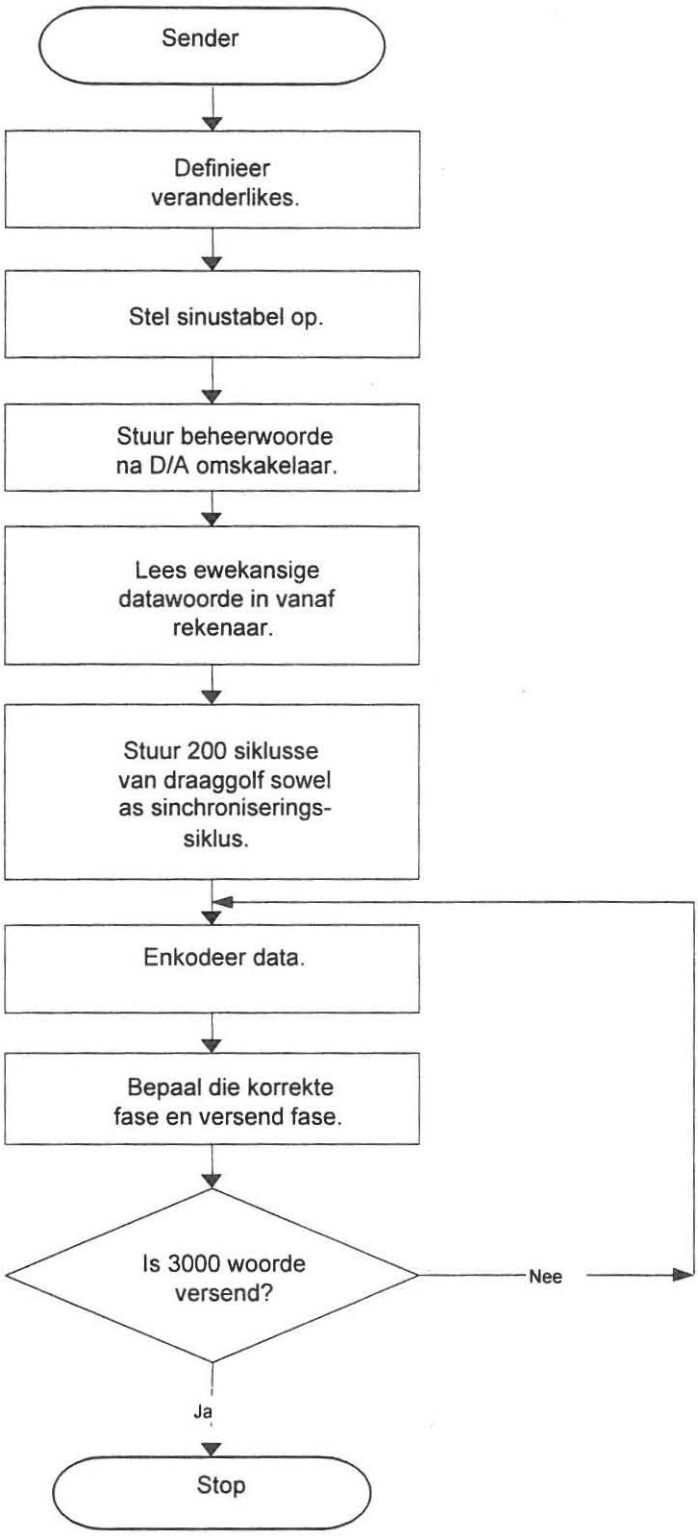
5.1 Beheersagteware

Die beheersagteware beheer die sender en ontvangerkaarte en is in Pascal geskryf. Dit verrig die volgende funksies.

- Die sender se sagteware word deur die beheersagteware vanaf die gasheerrekenaar na die DSP (sender) kaart oorgedra.
- Die ontvanger se sagteware word vanaf die gasheerrekenaar na die DSP (ontvanger) kaart oorgedra.
- Ewekansige datawoorde word deur die beheersagteware gegenereer en aan die senderkaart oorgedra.
- Ontvangde datawoorde word vanaf die ontvangerkaart na die gasheerrekenaar oorgedra.
- Ontvangde data word met versende data vergelyk ten einde die aantal bisfoute te bepaal.

5.2 Sender sagteware

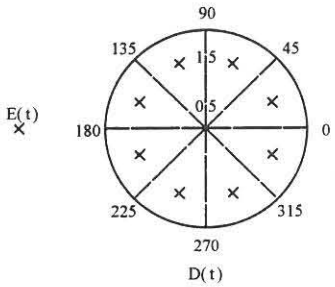
Hier volg 'n beskrywing van die sender sagteware wat in saamsteltaal geskryf is. Die sagteware word aan die hand van die vloeidiagram in Figuur 5-1 bespreek.



Figuur 5-1: Blokdiagram van die sender.

Eerstens word al die veranderlikes wat gebruik word gedefinieer en die nodige registers opgestel.

Vervolgens word die sinustabelle van die draer sowel as die gemoduleerde sein opgestel. Hierdie tabelle bevat elk agt monsters van 'n sinusgolf. Uit die tabel in Figuur 5-2 kan gesien word dat daar 'n faseverskil van 22.5° tussen die draer en die eerste simbool van die gemoduleerde sein bestaan.



Figuur 5-2: 8-Fases van sinustabel vir gemoduleerde sein.

Draer	Sintabel
0°	22,5°
45°	67,5°
90°	112,5°
135°	157,5°
180°	202,5°
225°	247,5°
270°	292,5°
315°	337.5°

Beheerwoorde word vervolgens vanaf die DSP-kaart na die digitaal-na-analoog omskakelaar gestuur. Hierdie beheerwoorde bepaal die monstertempo sowel as die tipe filter wat gebruik word. Die monstertempo van die digitaal-na-analoog omskakelaar is 3200 monsters per sekonde.

Nadat die digitaal-na-analoog omskakelaar opgestel is word 1000 datawoorde van 16 bisse elk ingelees vanaf die rekenaar en in die geheue van die sender DSP-kaart gestoor.

Sodra die sender die volledige stel datawoorde vanaf die gasheerrekenaar ontvang het word 200 volle siklusse van die ongemoduleerde draaggolf versend. By die ontvanger word die sinus sowel as die cosinus tabelle wat vir demodulasie gebruik gaan word vanaf hierdie draaggolf afgelei.

Nadat die draer vir 200 siklusse gestuur is, word daar 'n sinchroniseringsarsie gestuur. Hierdie sarsie het die doel om aan die ontvanger aan te dui wanneer daar wel data gestuur sal word, sowel as om die ontvanger te toets ten einde vas te stel of die sinus en cosinustabelle in die ontvanger reg opgestel is.

Die sarsie is as volg saamgestel. Daar word agt siklusse van 202.5° versend gevolg deur agt siklusse van 22.5°. Hierdie sarsie word 40 keer herhaal. Die ontvanger sal slegs sinchroniseer indien al hierdie fases korrek ontvang is. Sodra die sender en ontvanger gesinchroniseer het word 'n beheersein aan die gasheerrekenaar deur die ontvanger gestuur en die transmissie van data neem 'n aanvang. Indien die sender en ontvanger nie binne 'n sekere tyd korrek sinchroniseer nie, herstel die gasheerrekenaar die stelsel.

Indien die ontvanger reg gesinchroniseer het word die data in die sender ge-enkodeer. Die fase van die eerste ge-enkodeerde driebis waarde word dan bepaal en agt monsters word teen die korrekte fase ten opsigte van elke tribis versend.

Die totale hoeveelheid data wat per toets versend is, is 30000 datawoorde van 16 bisse elk. Hierdie toets is vir verskillende seinruisverhoudings herhaal.

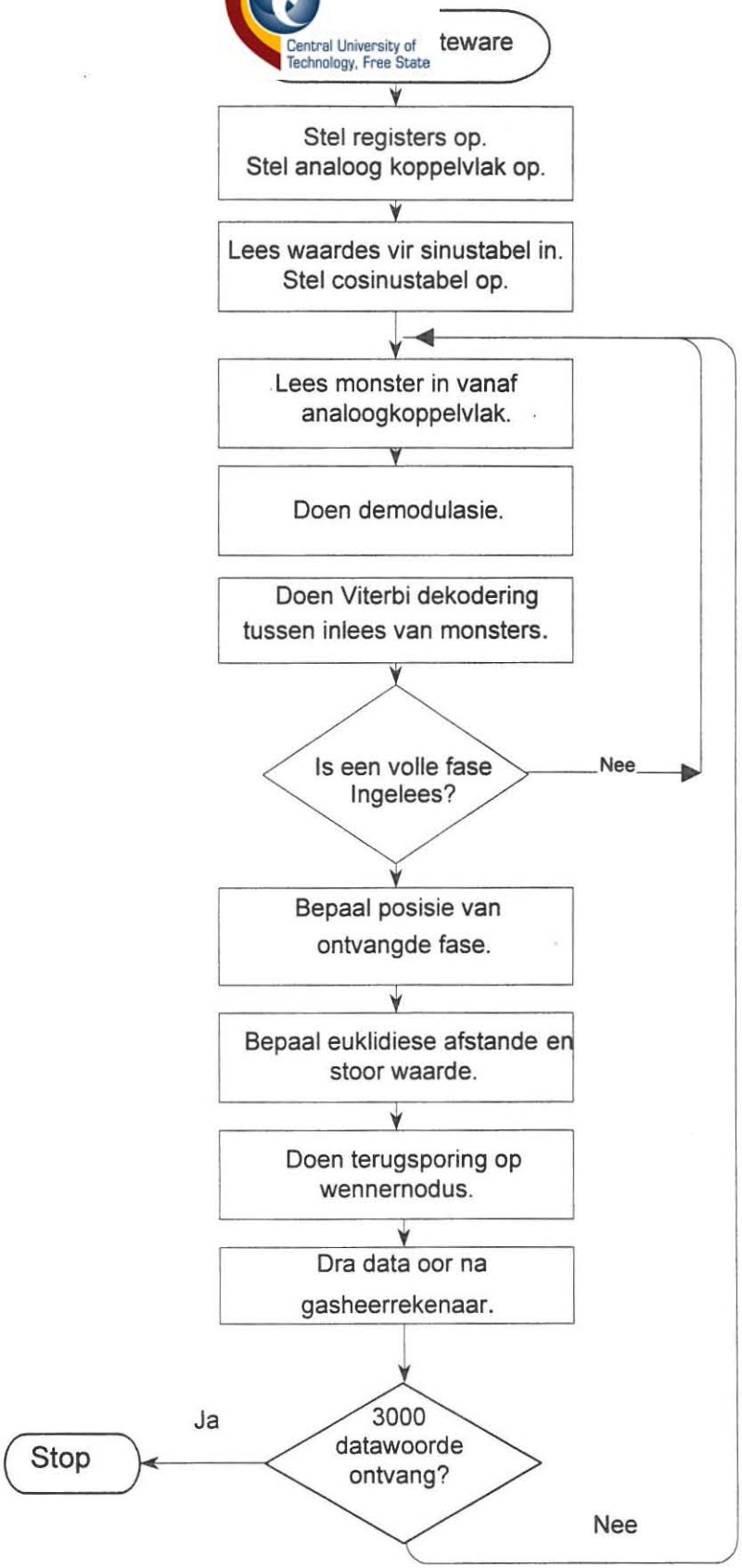
5.3 Ontvanger sagteware.

Die ontvanger se sagteware word aan die hand van die vloeidiagram in Figuur 5-3 bespreek.

Eerstens word die analoog koppelvlak sowel as die werkregisters opgestel[5]. Die sender stuur 200 siklusse van die draaggolf. By die ontvanger word bepaal wanneer die eerste monster van 'n sinusgolf ontvang word. Dit word gedoen deur te kyk wanneer die monsterwaardes vanaf negatief na positief beweeg. Hierdie monster en die daaropvolgende sewe monsters word dan in die sinustabel ingelees. Indien die sinustabel nie korrek is nie sal die ontvanger nie die sinchroniseringsiklus korrek ontvang nie en die beheersagteware sal die stelsel herstel. Die toets vir die korrektheid van die sinustabel is dus of die sinchroniseringsiklus korek ontvang word. Die cosinustabel word vanaf die sinustabel afgelei. Daar word een monster op 'n slag vanaf die analoogkoppelvlak ingelees. Die ingeleesde waarde word dan verwerk voor daar 'n volgende monster ingelees word. Ten einde demodulasie van die sein te bewerkstellig word die ingeleesde waarde nou met die ooreenstemmende waarde in die sinustabel sowel as die cosinustabel vermenigvuldig[6]. Hierdie sinus en cosinusvermenigvuldigde waardes dui die x en y posisies van die ingeleesde simbool aan. Omdat die prosesseringstyd beperk is kan al die verwerking van die data nie geskied nadat al agt monsters van 'n fase ingelees is nie. Dit het tot gevolg dat van die verwerking tussen die inlees van monsters moet geskied.

---

[5] Die opstelling van die analoogkoppelvlak word in paragraaf 4.3.2 bespreek.
[6] Die wiskundige voorstelling word in paragraaf 3.2.7 bespreek.



Figuur 5-3: Vloeikaart van ontvanger agteware.

Sien die bespreking van Viterbi-dekodering in paragraaf 3.3 en ten einde meer duidelikheid oor die volgende deel van die sagteware te verkry. Die euklidiese-afstande van die vorige nodusse tot by die huidige nodus word vervolgens geneem en by die totale afstande van die vorige nodusse getel. Dan word die sprong met die kortste totale pad as die wenner geneem en gestoor.

Nadat die wenner bepaal is word die totale afstand, sowel as die oorspronklike nodus in 'n matriks gestoor. Hierdie proses word vir elke nodus herhaal en word gedurende die tyd tussen die inlees van die monsters gedoen.

Die bepaling van die euklidiese afstand is noodsaaklik ten einde Viterbi-dekodering deur middel van sagte besluitneming te realiseer[7]. Indien die afstand tussen twee punte op 'n plat vlak verlang word word dit as volg bepaal.

$$a = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \qquad (5\text{-}1)$$

Waar:

a = afstand tussen die twee punte

$x1$ en $y1$ = die x en y koördinate van die een punt.

$x2$ en $y2$ = die x en y koördinate van die tweede punt.

[7] Sien paragraaf 3.3.2 vir meer duidelikheid.

Omdat die berekening van die vierkantswortel heelwat prosesseringstyd in beslag neem en dit nie nodig is om die presiese afstand te bepaal nie, maar slegs die verhouding tussen die afstande, is die volgende formule gebruik.

$$a = (x1 - x2)^2 + (y1 - y2)^2 \qquad (5\text{-}2)$$

Omdat die agt finale afstande vanaf die agt moontlike fases nou bekend is kan die kortste totale afstand nou geneem word en daar kan teruggespoor word. Die data wat verkry word aan die begin van die pad word as die korrekte data geneem en oorgedra na die gasheerrekenaar. Tabel 5-1 toon die verloop van die basiese take van die verskillende sagtewarekomponente.

Tabel 5-1 Aanduiding van watter prosedures op watter staduim deur die verskillende sagtewarekomponente uitgevoer word.

Beheersagteware	**Sendersagteware**	**Ontvangersagteware**
Beheersagteware word in gasheerrekenaar gelaai.		
Beheersagteware laai sender en ontvanger-sagteware in DSP-kaarte.		
Beheersagteware stuur 1000 datawoorde na sender.	Sender ontvang datawoorde en stoor in geheue.	Ontvanger wag vir sinchroniseringsiklus vanaf sender.
	Sender stuur sinchroniseringsiklus.	Ontvanger lei sinus en cosinus tabelle vanaf sinchroniseringsiklus af en sinchroniseer met sender.
Beheersagteware wag vir ontvanger om ontvangde datawoorde na gasheerrekenaar oor te plaas.	Sender versend datawoorde.	Ontvanger ontvang datawoorde en stoor in geheue.
Beheersagteware ontvang datawoorde en plaas dit in geheue.		Ontvanger plaas 1000 datawoorde oor na gasheerrekenaar.

tyd ↓

Die ontvangde data word met die oorspronklike data vergelyk en die aantal foute word bepaal.		

5.4 Opsomming

In hierdie hoofstuk is die hoof prosedures van die verskillende sagteware komponente bespreek. Die sinchronisering van die sender en ontvanger is behandel en daar is gekyk na die rol van die beheersagteware in die proses.

6 Evaluering van datasender en ontvanger.

Hier volg 'n beskrywing van die verskillende komponente van die toetsstelsel. Die eksperimentele opstelling word in Figuur 4-1 aangetoon.

Die spesifikasies van die kommunikasiestelsel is as volg.

Draagfrekwensie = 400Hz

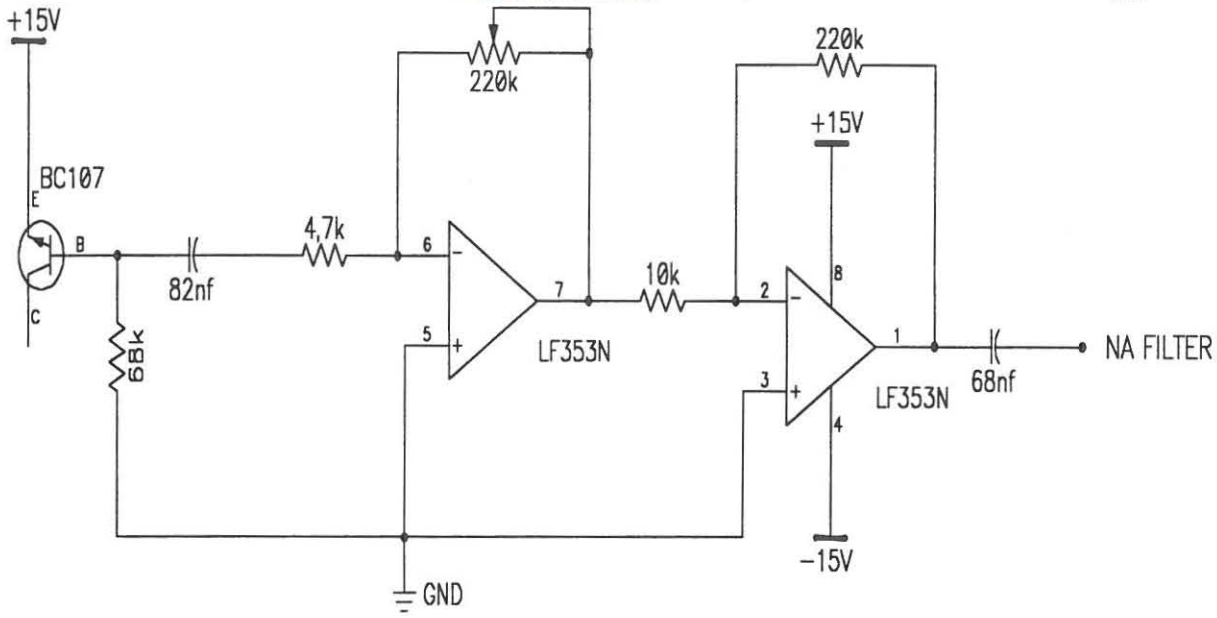
Bistempo = 1200 bisse per sekonde

Simbooltempo = 400 simbole per sekonde

Datatempo = 800 bisse per sekonde

6.1 Die ruisgenerator

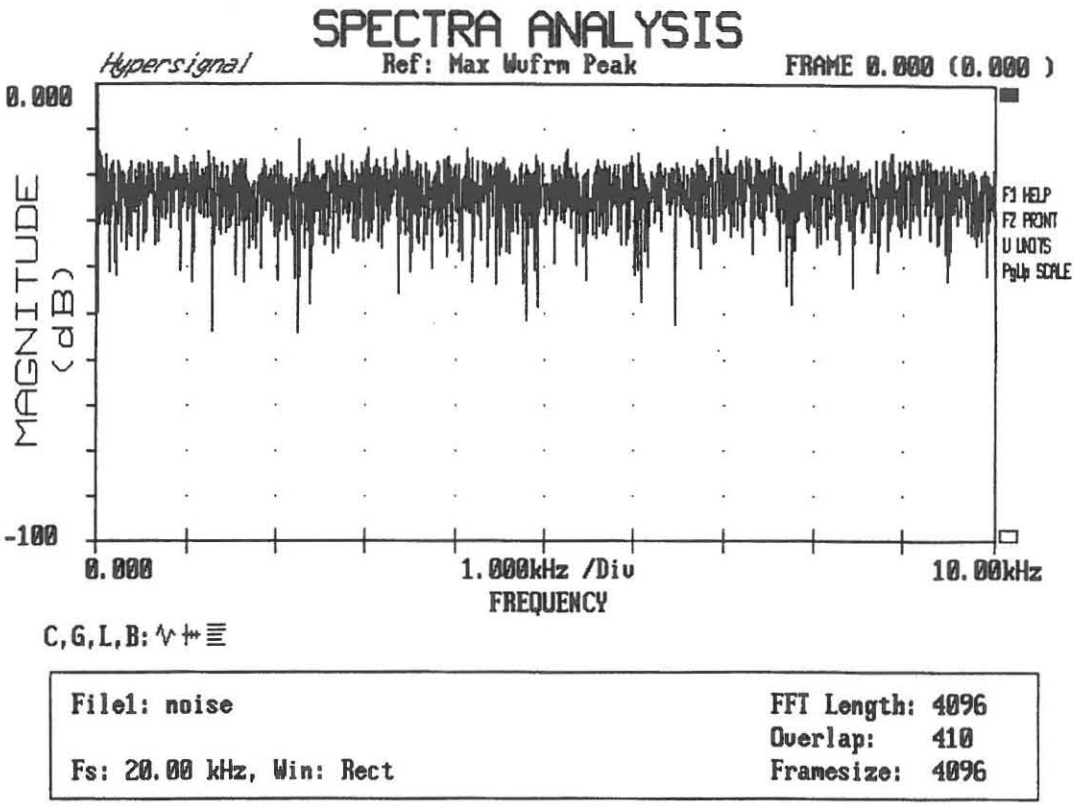
Ten einde die stelsel se werkverrigting teen verskillende seinruisverhoudings te kon bepaal moes daar ruis by die sender se uitset gevoeg word. Die stroom wat oor die teenvoorgespanne basis-emittor voegvlak van 'n BC107 transistor beweeg veroorsaak 'n oneweredige aankoms van elektrone op die voegvlak en sodoende word haelruis gegenereer. Omdat hierdie ruissein baie klein is, is dit nodig om dit te versterk. Twee operasionele versterkers in kaskade is hiervoor gebruik. Die stroombaan van die ruisgenerator word in Figuur 6-1 aangetoon.



Figuur 6-1: Stroombaan van die ruisbron.

6.2 Ruisfilter

Die ruisgenerator se uitset is konstant oor 'n frekwensiespektrum van 0 tot 10kHz. Dit is voldoende aangesien die stelsel ontwerp is vir 'n kanaal met 'n 3kHz bandwydte en 'n laagdeurlaat ruisfilter met 'n afsnyfrekwensie van 2.4 kHz gebruik is. Figuur 6-2 dui die gemete frekwensiespektrum van die ruis tussen 0 en 10kHz aan.



Figuur 6-2: Frekwensiespektrum van ruis.

Die ruisgenerator se uitset strek oor 'n wyer band as die bandwydte van die senderuitset. Die uitset van die ruisgenerator is derhalwe met 'n laagdeurlaatfilter gefilter voordat dit by die datasein gevoeg is. In die ontwerp van die filter is 'n 3dB afsnyfrekwensie van 2500Hz verlang. Die filter wat gebruik is, is 'n 6de orde laagdeurlaat Butterworth filter met 'n verswakking van 36dB per oktaaf. Die boonste afsnyfrekwensie is gekies as 10% onder die -3dB punt.

Die afsnyfrekwensie fp word as volg bereken [15, p. 3.45].

$$fp = \frac{1}{2 \cdot \pi \cdot R \cdot C} \qquad (6\text{-}1)$$

Die wins van die kring in Figuur 6-4 word as volg bereken.

$$A = \left(\frac{R7}{R8} + 1\right) \bullet \left(\frac{R10}{R11} + 1\right)$$

=3.85 (11.6dB)

Vir die verlangde bandwydte 0 tot 2500Hz is:

fp =10% laer as 2500Hz

$$fp = 2500 \bullet 0.9 = 2250Hz$$

Indien C geneem word as 10nF dan is

$$R = \frac{1}{fp \cdot 2 \cdot \pi \cdot C}$$

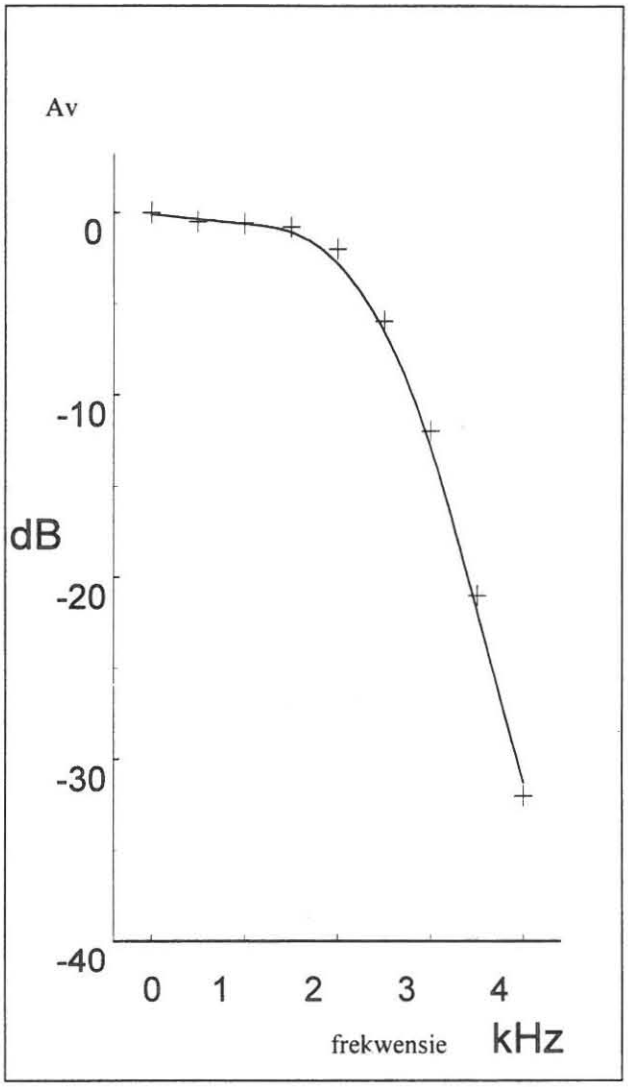
=7k073

Die naaste E12 waarde is 6k8 . Dit lewer 'n aangepaste afsnypunt van:

$$\mathrm{fp} = \frac{1}{2 \cdot \pi \cdot \mathrm{R} \cdot \mathrm{C}}$$

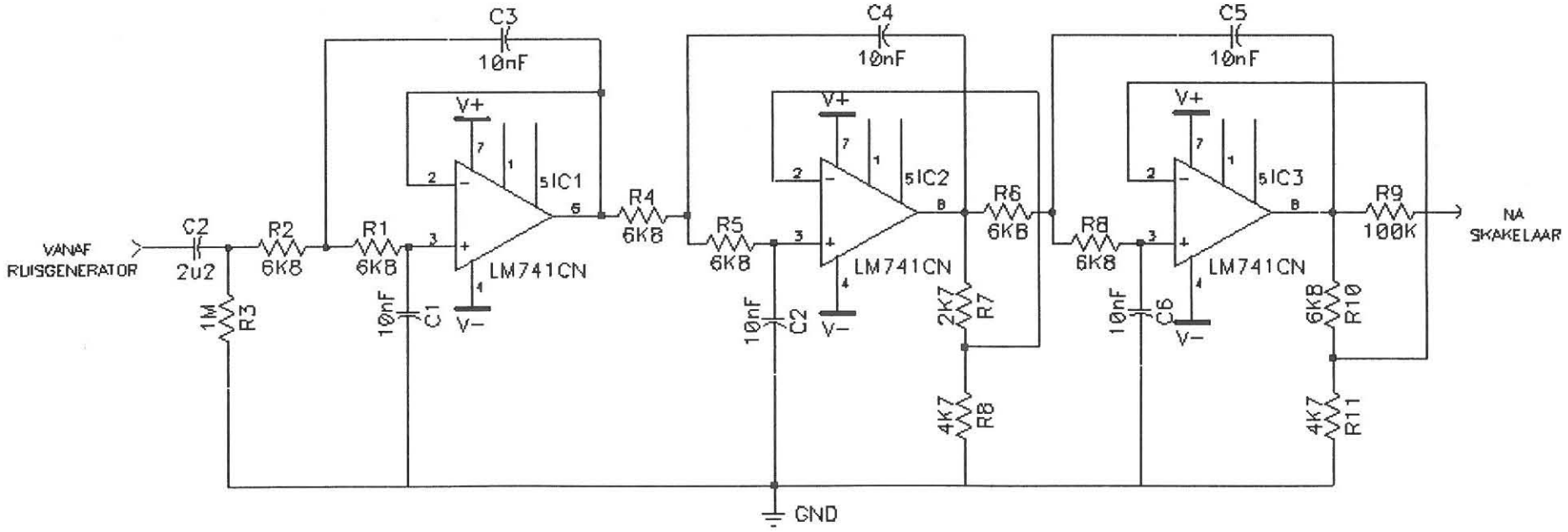
=2.341kHz

In die praktiese evaluering het hierdie waardes bevredigende resultate gelewer. Die karakeristieke weergawe van die ruisfilter, soos gemeet, word in Figuur 6-4 aangetoon.



Figuur 6-3: Frekwensieweergawe van ruisfilter.

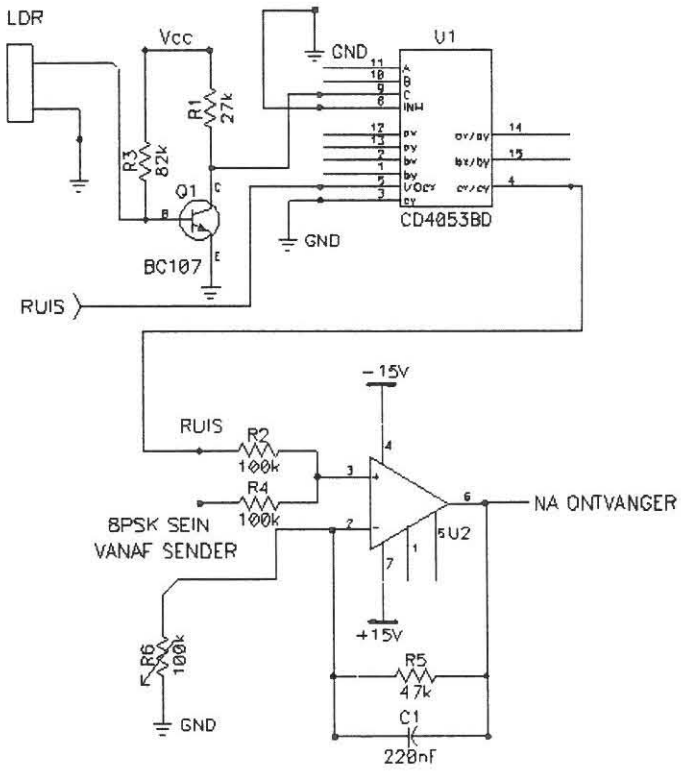
Die stroombaan van die ruisfilter word in Figuur 6-4 aangetoon.



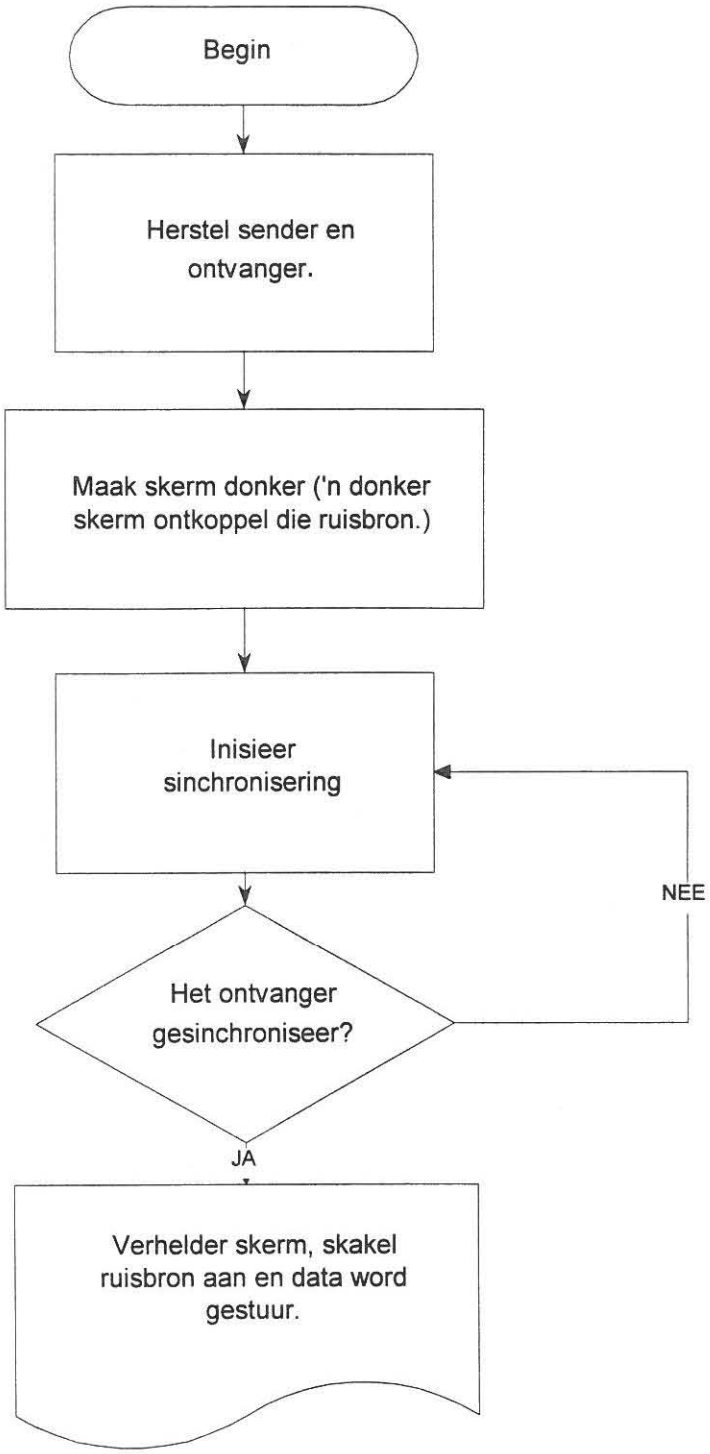
Figuur 6-4: Stroombaan van ruisfilter.

6.3 Ruisskakelaar

Daar is gevind dat die stelsel nie suksesvol by hoë ruisvlakke kon sinchroniseer nie. Die stelsel is toe so aangepas dat die ruis eers na sinchronisasie bygevoeg is. Dit is bewerkstellig deur sagteware wat die rekenaar skerm verdonker terwyl die sinchronisasie plaasvind, en sodra die ontvanger op die sender gesinchroniseer is, word die skerm verhelder, en die ruis word by die sender se uitset gevoeg. Op die skerm is 'n ligsensitiewe weerstand geplaas wat deur middel van 'n skakelstroombaan die ruis koppel of ontkoppel. Figuur 6-5 dui die stroombaan van die ruisskakelaar aan.



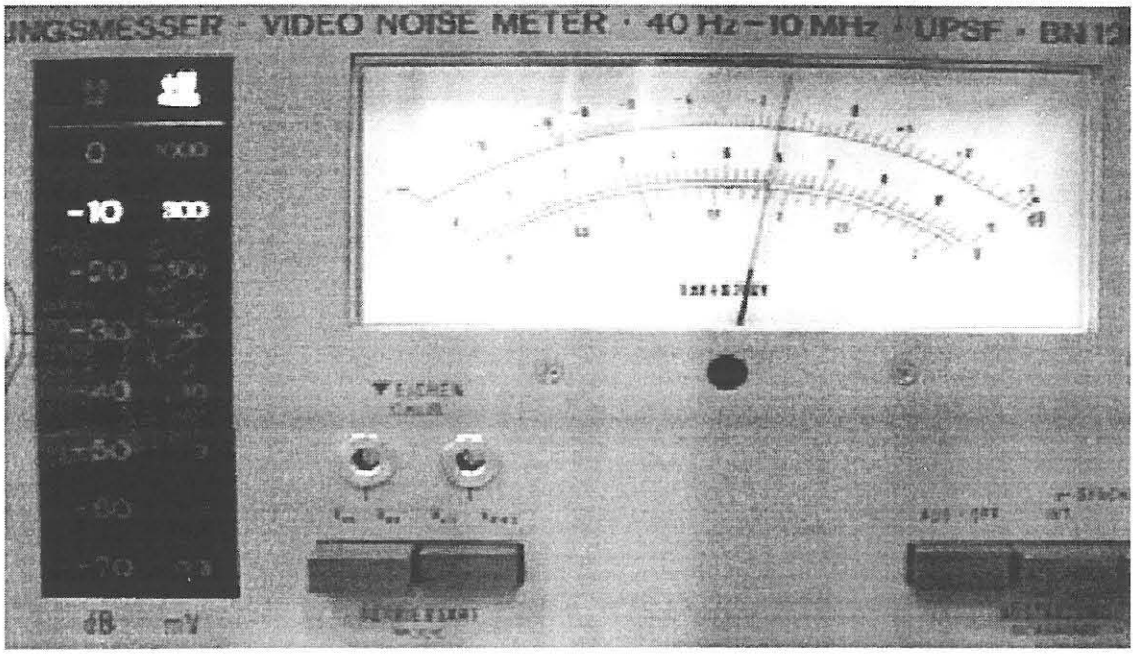
Figuur 6-5: Stroombaan van ruisskakelaar.



Figuur 6-6: Vloeikaart van ruisbeheer proses.

6.4 Ruismeter

Daar is van 'n hoë kwaliteit Rohde & Schwarz ruismeter[8] gebruik gemaak om akkurate sein en ruislesings te verseker.



Figuur 6-7: Ruismeter.

6.5 Evaluering van sender.

Die evaluering van die sender het die volgende behels.

- Die uitset is deur middel van 'n ossilloskoop gemonitor.

- Daar is 'n spektrumanalise op die frekwensiespektrum van die sein gedoen.

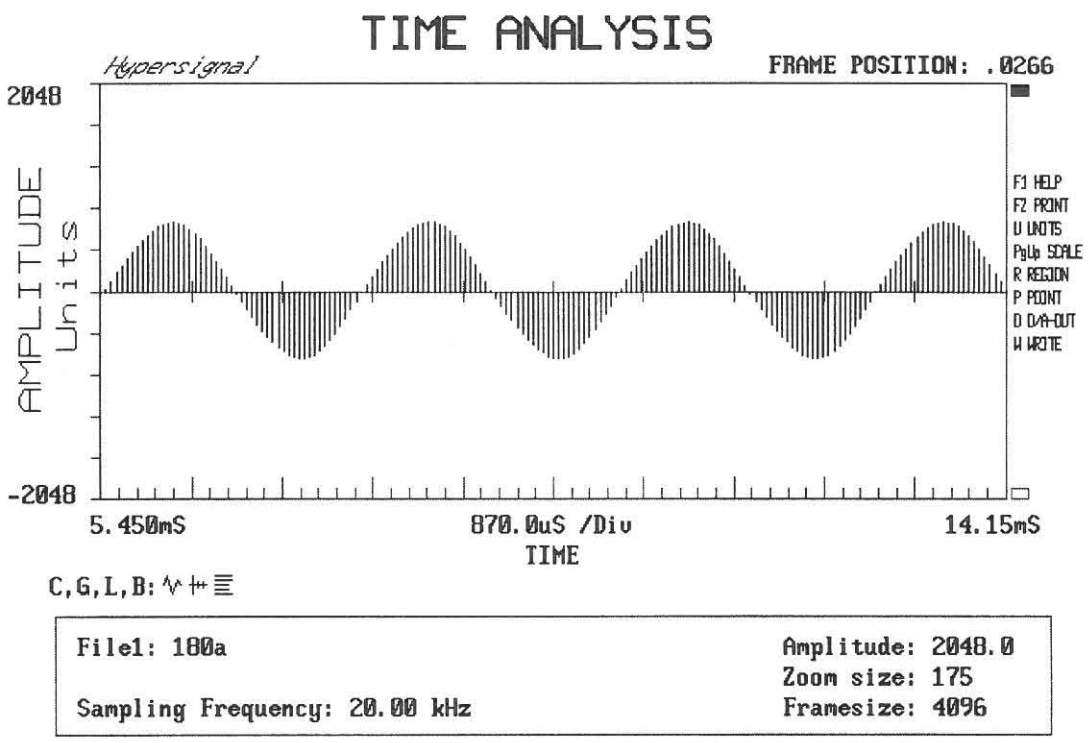
[8] Model: BN120312

6.5.1 Metode van bemonstering.

Ten einde die sender te kon evalueer ten opsigte van werkverrigting, is verskillende uitsette eers bemonster en daarna ontleed. Die bemonstering is met behulp van 'n PC30 kaart teen 'n monstertempo van 20000 monsters per sekonde gedoen. Nadat die monsters geneem is, is dit met behulp van 'n sagteware pakket (Hypersignal[9]) ontleed.

6.5.2 Ontleding van draaggolf.

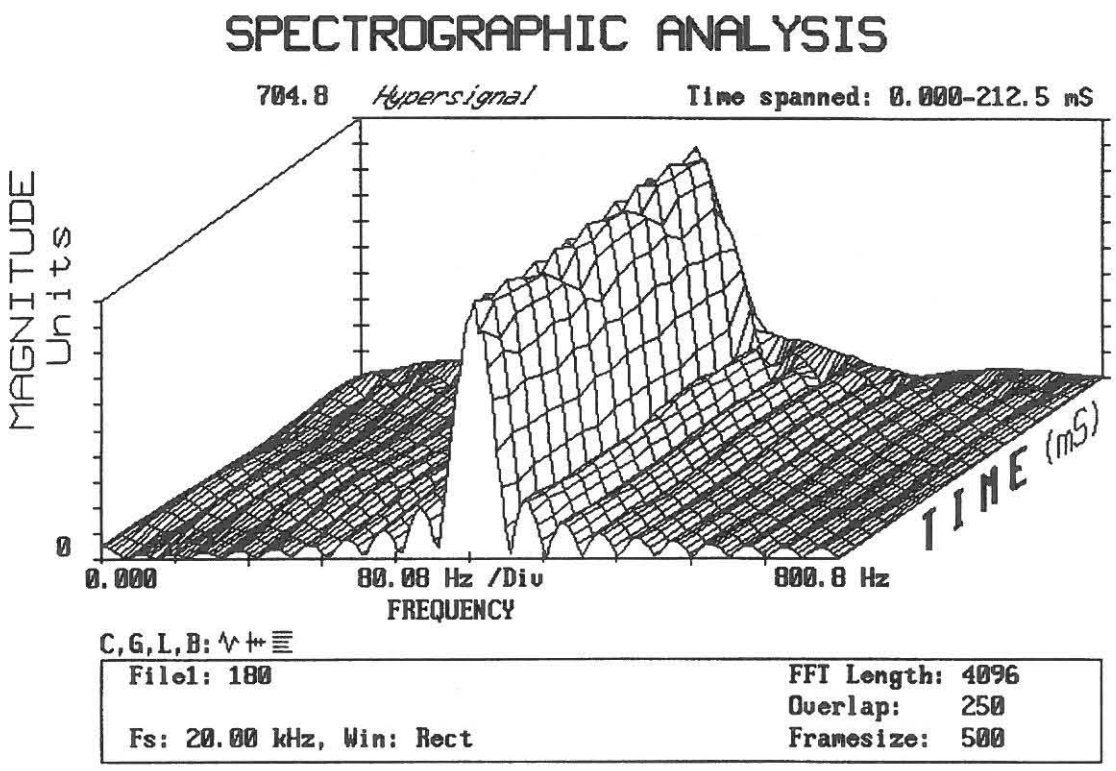
Vir die doel van die ontleding van die draaggolf is die sender so verander dat slegs die draaggolf gestuur is. Die uitset is gemonster en Figuur 6-8 dui die golf aan soos deur Hypersignal vertoon.



Figuur 6-8 Bemonsterde draaggolf.

[9] Hypersignal is die eiendom van Signalogic met kopiereg op die program.

Vir die bepaling van die frekwensiesamestelling is die spektrum ontleed. Figuur 6-9 dui die spektrum van die draaggolf. 'n Spektrale uitset van 400Hz (as draagfrekwensie) is verkry.

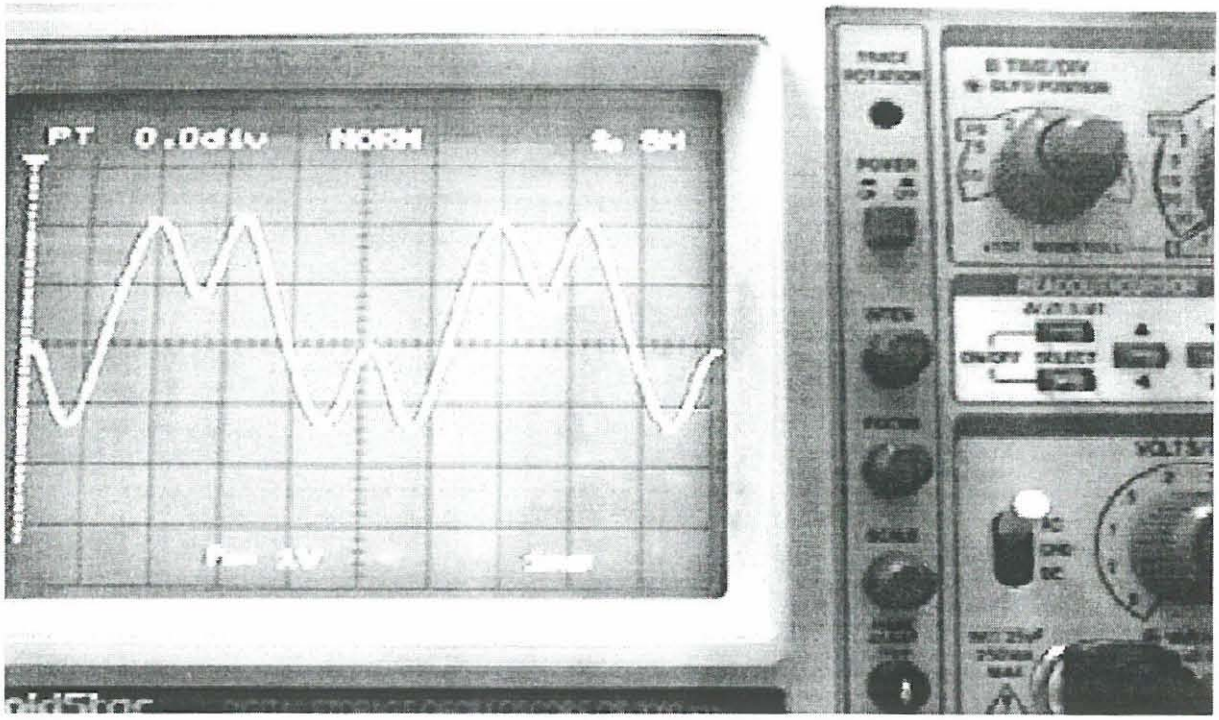


Figuur 6-9: Frekwensiespektrum van 400Hz draaggolf.

6.5.3 Ontleding van die leersiklus.

Om sinchronisasie te bewerkstellig stuur die sender eers 'n leersiklus aan die ontvanger. Hierdie siklus bestaan uit 180 grade faseveranderings ten opsigte van elke opeenvolgende simbool.

Figuur 6-10 toon 'n gedeelte van die leersiklus aan soos gesien op 'n ossilloskoopskerm.

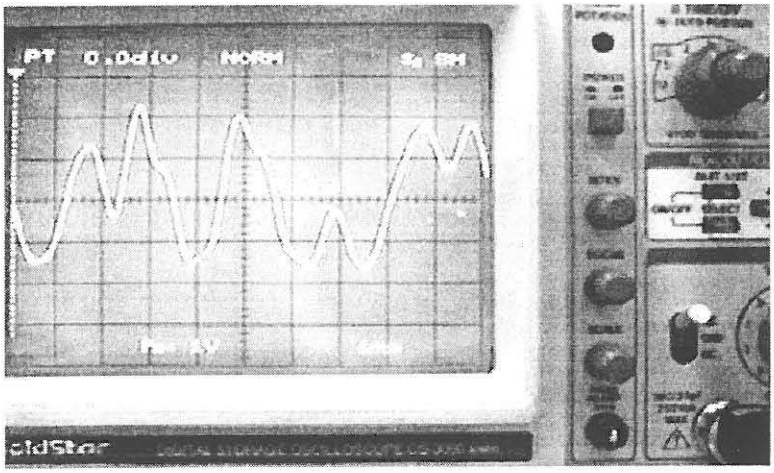


Figuur 6-10: 8-PSS sein met 180 grade faseverskuiwing soos gebruik in sinchronisering van die stelsel.

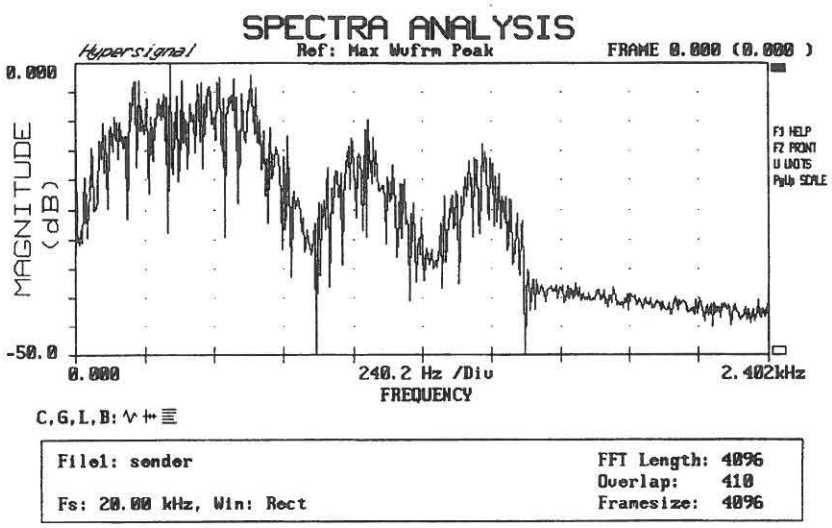
6.5.4 Ontleding van die gemoduleerde sein.

Die tydweergawe van die 8-PSS gemoduleerde sein, soos gesien op 'n ossilloskoopskerm, word in Figuur 6-11 aangetoon. Die gemoduleerde sein bevat prominente pieke op 1000Hz en 1400Hz[10]. Omdat die draer 400Hz is, en die sein gemoduleer is, word die frekwensieverspreiding aan weerskante van 400Hz verwag. Die volgende piek kom op 1000Hz voor. Dit kan verklaar word as die derde harmoniek van die maksimum frekwensie van die moduleersein bo die 400Hz draer. Die laaste komponent kom op 1400Hz voor wat die vyfde harmoniek van die maksimum frekwensieverandering in die moduleersein voorstel.

[10] Hierdie pieke word in Figuur 6-12 aangetoon.



Figuur 6-11 Tipiese gemoduleerde 8-PSS sein.



Figuur 6-12 Frekwensiespektrum van die gemoduleerde sein.

Evaluering van ontvanger.

Die volgende toetse is uitgevoer.

- Deur middel van 'n Pascal program is 1000 ewekansige datawoorde gegenereer. Die datawoorde het elk uit 16 bisse bestaan.

- Die 1000 ewekansige datawoorde is dan vanaf die gasheerrekenaar na die DSP-sender oorgestuur.

- Die sender het die 1000 woorde drie keer versend en die ontvanger het dit so ontvang.

- Die aantal bisse versend is dus die 1000 datawoorde vermenigvuldig met drie[11] en dan vermenigvuldig met 16[12]. Die totaal is dus 48000 bisse.

- Hierdie toets is 10 keer herhaal vir verskillende seinruisverhoudings. Die totale aantal bisse getoets, teen elke seinruisverhouding is 480000.

- Na ontvangs van elke stel bisse, is dit met 'n toetsprogram (wat in Pascal geskryf is) getoets. Die program vergelyk die bisse van die versende datawoorde met die van die ontvangde datawoorde. Die program gee die aantal bisfoute as 'n uitset.

6.6.1 Resultate

By die evaluering van die ontvanger is die resultate soos in Tabel 1 verkry.

Die **BER** is die persentasie databisfoute soos ontvang.

Tabel 1: Gemete waardes van die BER ten opsigte van die ontvangers met en sonder Viterbi dekodering.

[11] Die sender versend hierdie data drie keer.
[12] Elke woord bestaan uit 16 bisse.

E/No (dB)	Trellis en Viterbi (BER)	Trellis sonder Viterbi (BER)
18.061	0	0
15.563025	0	4.167E-0
13.624825	0	4.167E-0
12.041	0	0.0001229
10.702264	0	0.0008771
9.5424251	0	0.0038396
8.5193746	0	0.0074438
7.6042248	0	0.0089667
6.0205999	0	0.0373
4.6816641	7.083E-0	0.0343063
3.5218252	0.0001938	
2.4987747	0.000975	
1.5836249	0.0048271	
0.7557712	0.0134417	
0	0.0238604	

Die seinruisverhouding is as volg bereken [13, p.133].

$$E / No(dB) = 10 \cdot \log(\frac{Vs^2}{Vn^2}) \qquad (6\text{-}1)$$

Waar: Vs =seinspanning

Vn =ruisspanning.

6.6.2 Gevolgtrekking

As gevolg van die kodewins van die trellis ge-enkodeerde 8-PSS stelsel word die werkverrigting daarvan met 'n ongekodeerde 4-PSS stelsel vergelyk [4, p. 425]. Die teoretiese werkverrigting van die 4-PSS stelsel word deur uitdrukking 6-3 weergegee [4, p. 317].

$$Pe = erfc\left(\sqrt{\frac{E}{No}}\sin\left(\frac{\pi}{M}\right)\right) \qquad (6\text{-}3)$$

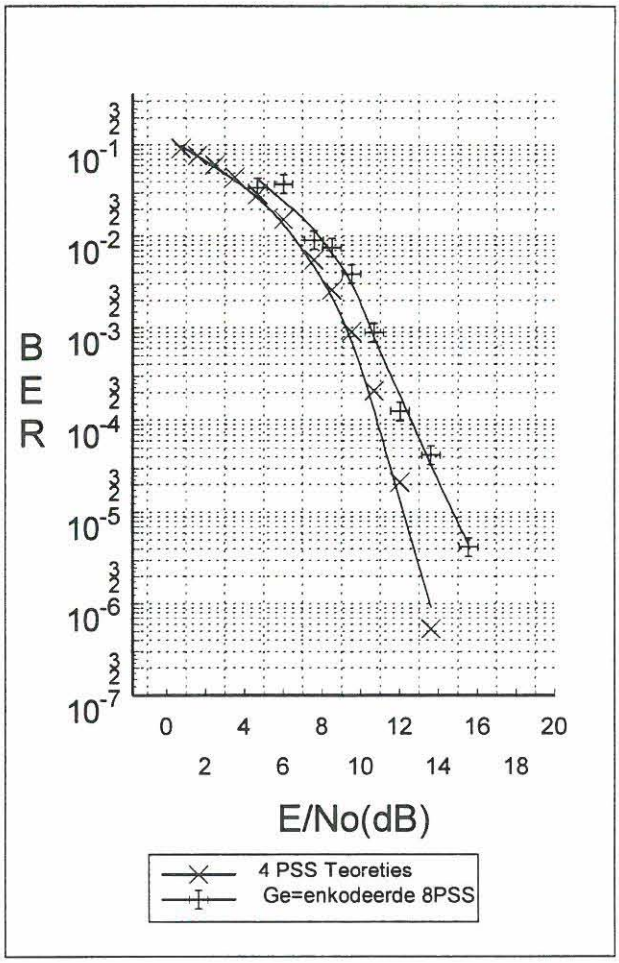
Waar: M = aantal simbole.

E = seindrywing.

No = ruisdrywing.

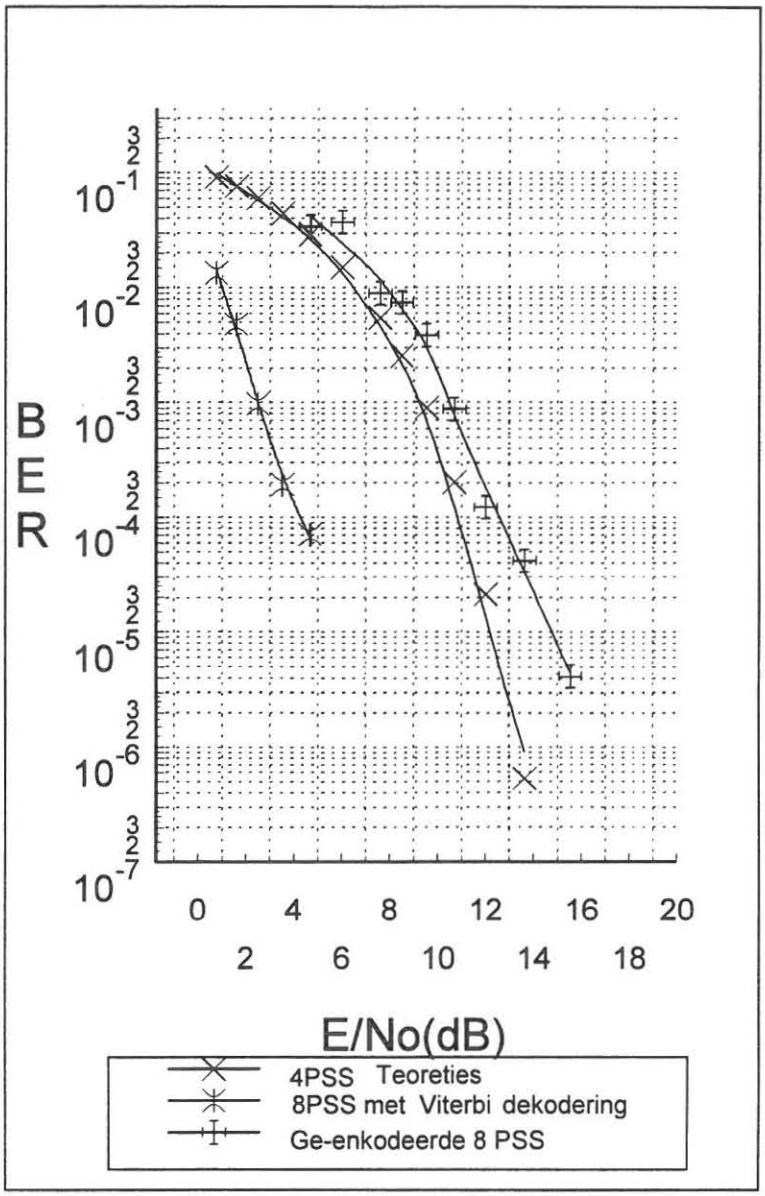
In Figuur 6-13 word die gemete uitset van die eksperimentele ge-enkodeerde 8-PSS stelsel gestip teen die teoretiese werkverrigting van 'n 4-PSS stelsel.

Soos verwag is die werkverrigting van die praktiese ge-enkodeerde 8-PSS swakker as die van die teoreties bepaalde waardes van die 4-PSS stelsel. Dit kan toegeskryf word aan die nie-perfekte eksperimentele toestande. By 'n bisfouttempo van 10E-2 is daar slegs 'n 1 dB verskil tussen die gemete ge-enkodeerde 8-PSS en die teoretiese 4-PSS stelsel. By 'n bisfouttempo van 10E-4 is daar egter 'n 2 dB verskil. Volgens Haykin is daar 'n 5dB verskil tussen die werkverrigting van ongekodeerde 4PSS en ongekodeerde 8-PSS [4, p. 336]. Daar kan dus afgelei word dat die ge-enkodeerde 8-PSS sein wel 'n kodewins besit relatief tot 'n ongekodeerde 8-PSS sein.



Figuur 6-13 Werkverrigting van ge-enkodeerde 8-PSS en teoretiese 4-PSS.

'n Verdere wins word verwag indien Viterbi met sagte besluitneming in die ontvanger toegepas sou word. Dit kan dan ook dan duidelik in Figuur 6-14 waargeneem word - waar die teoretiese 4-PSS, die ge-enkodeerde 8-PSS en die ge-enkodeerde 8-PSS met Viterbi met mekaar vergelyk word. Die ge-enkodeerde 8-PSS stelsel met Viterbi dekodering het 'n 6.5 dB wins relatief tot die teoretiese waardes van die 4-PSS stelsel.



Figuur 6-14 Werkverrigting van 4PSS, 8-PSS met Viterbi dekodering en ge-enkodeerde 8-PSS stelsels.

6.7 Opsomming

In die uitvoering van die projek is gevind dat Viterbi dekodering wel in DSP implementeerbaar is en dat bevredigende resultate ten opsigte van die aantal bisfoute vir verskillende seinruisverhoudings verkry is. Daar is gevind dat die werkverrigting van 'n trellis ge-enkodeerde 8-PSS sein vergelykbaar is met die

van 'n ongekodeerde 4PSS sein. Dit dui op die kodewins van die trellis ge-enkodeerde sein. In die implementering van die ge-enkodeerde 8-PSS stelsel, waar van Viterbi dekodering gebruik gemaak is, is 'n wins van 6.5dB relatief tot 4-PSS verkry. Dit is nie onrealisties indien dit met 'n wins van 5.2 dB, soos deur Haykin vir 'n 2/3 enkodeerder aangegee, vergelyk word nie [4, p. 429].

7 Samevatting

7.1 Doelwitte

In die uitvoering van die projek is die volgende doelwitte bereik.

- Trelliskodemodulasie sowel as Viterbi-dekodering is bestudeer.
- 'n Trellis-enkodeerder en 'n 8-PSS datasender is gesimuleer.
- 'n 8-PSS sender en ontvanger wat gebruik maak van viterbi-dekodering is ontwikkel.
- Die sender sowel as die ontvanger is in DSP geïmplementeer deur van SIG-56 DSP ontwikkelingskaarte gebruik te maak.
- Die trelliskodemodulasie sowel as die DSP implementering daarvan is geevalueer en bevredigende resultate is verkry.

7.2 Inhoud van hoofstukke

Hoofstuk 1 is inleidend van aard.

In hoofstuk 2 is 'n teoretiese oorsig van digitale kommunikasie gegee, en verskillende enkoderings en modulasietegnieke is bespreek.

Hoofstuk 3 handel oor verskillende digitale koderingstelsels. In die hoofstuk word die verskille tussen blok en konvolusiekodes uitgewys en die Viterbi dekoderingsbeginsel bespreek.

In hoofstuk 4 word die uiteensetting van die hardeware gedoen. Die eksperimentele opstelling en elke hardeware komponent word bespreek.

In hoofstuk 5 word die hoof prosedures van die verskillende sagteware komponente bespreek. Die sinchronisering van die sender en ontvanger is behandel en daar is gekyk na die rol van die beheersagteware in die proses.

In hoofstuk 6 is die trelliskodestelsel met Viterbi dekodering ge-evalueer. Waar van Viterbi dekodering gebruik gemaak is, is 'n wins van 6.5dB relatief tot 4-PSS verkry. Dit is nie onrealisties indien dit met 'n wins van 5.2 dB, soos deur Haykin vir 'n 2/3 enkodeerder aangegee, vergelyk word nie [4, p. 429].

Bylae A: Resultate van aantal bisfoute soos eksperimenteel bepaal vir stelsel met Viterbi-dekodering.

S/No(dB)	18	15.5	13.6	12	10.5	9.5	8.5	7.5	6	4.5	3.5	2.5	1.5	0.75	0.2
Toets no.															
1	0	0	0	0	0	0	0	0	0	0	8	34	356	514	1199
2	0	0	0	0	0	0	0	0	0	0	4	24	391	590	1005
3	0	0	0	0	0	0	0	0	0	0	10	40	103	529	960
4	0	0	0	0	0	0	0	0	0	16	6	3	324	929	1433
5	0	0	0	0	0	0	0	0	0	0	3	41	304	852	1084
6	0	0	0	0	0	0	0	0	0	0	6	28	155	602	1021
7	0	0	0	0	0	0	0	0	0	4	20	49	83	882	980
8	0	0	0	0	0	0	0	0	0	14	18	56	140	414	1100
9	0	0	0	0	0	0	0	0	0	0	6	110	232	625	1274
10	0	0	0	0	0	0	0	0	0	0	12	83	229	515	1397
Totale aantal bisfoute Uit 480000 bisse versend	0	0	0	0	0	0	0	0	0	34	93	468	2317	6452	11453

Tabel 2: Resultate van aantal bisfoute soos eksperimenteel bepaal vir stelsel met Viterbi-dekodering.

Bylae B: Resultate van aantal bisfoute soos eksperimenteel bepaal vir stelsel sonder Viterbi-dekodering.

S/No(dB)	18	15.5	13.6	12	10.7	9.5	8.5	7.5	6	4.5
Toets no.										
1	0	0	5	4	38	112	190	398	947	1394
2	0	0	4	3	33	127	356	388	925	1436
3	0	0	0	10	41	232	299	452	897	1751
4	0	0	3	6	48	211	321	347	1144	1640
5	0	0	0	3	34	198	540	534	1065	1855
6	0	0	5	2	53	168	426	684	1016	1686
7	0	0	0	6	51	178	361	372	955	1608
8	0	0	2	8	44	216	377	426	986	1665
9	0	0	0	6	47	211	319	319	1092	1536
10	0	0	1	11	32	190	384	384	1877	1896
Totale aantal bisfoute Uit 480000 bisse versend	0	0	20	59	421	1843	3573	4304	10904	16467

Tabel 3: Resultate van aantal bisfoute soos eksperimenteel bepaal vir stelsel sonder Viterbi-dekodering.

Bylae C: Beheer-sagteware

```
program Beheersagteware;
uses crt,dsplib4;

var j                                   :byte;
c                                       :char;
fout,chkout,chkin                       :integer;
baseaddress                             :word;
datafile1,datafile2,datafile3           :text;
data                                    :array[1..4000]of longint;
van_file                                :array[1..7200] of longint;
na_file                                 :array[1..4200] of longint;
i,p,m,q,r,k,qq                          :integer;
l                                       :longint;
d_in,e                                  :longint;
d_uit                                   :longint;
BaseS,BaseR,timeout                     :word;
sein,dat                                :text;

PROCEDURE DP_ReadFlag(BaseAddress,Timeout:word);
var  TimeoutCount:word;
     TimeoutError:boolean;
Begin
 TimeoutError:=false;
 TimeoutCount:=0;
 fout:=0;
 repeat
  TimeoutCount:=succ(timeoutCount);
  Delay(1);
  until ((port[BaseAddress+2] and 1)=1) or (TimeoutCount>Timeout);
 TimeoutError:=TimeoutCount>Timeout;
 If TimeoutError then
 begin
  write('!');
  fout:=1;
  { halt;   }
 end;
End;

PROCEDURE
Message(BackGColor,ShadowColor,MenuColor,BorderColor:byte;
                X1,Y1,X2,Y2:Integer);
VAR
 Hulp : Integer;

BEGIN
 TextBackground(BackGColor);
```

```
  ClrScr;
  Window(X1+1,Y1+1,X2+1,Y2+1);
  TextBackground(ShadowColor);
  ClrScr;
  Window(X1,Y1,X2,Y2);
  TextBackground(MenuColor);
  ClrScr;
  TextColor(BorderColor);
  Writeln(' É');
  For Hulp := 3 to (X2-X1-1) do
  Begin
    GotoXY(Hulp,1);
    Write('Í');
  End;
  Write('»');
  For Hulp :=1 to (Y2-Y1-1) do
  Begin
    GotoXY(1,Hulp+1);  Write(' º');
    GotoXY(X2-X1,Hulp+1);  Write('º');
  End;
  GotoXY(1,Y2-Y1+1);
  Write(' È');
  For Hulp := 3 to (X2-X1-1) do
  Begin
    GotoXY(Hulp,Y2-Y1+1);
    Write('Í');
  End;
  Write('¼');
End;
Begin

  for k:=1 to 7200 do
        begin
        van_file[k]:=0
        end;
  for k:=1 to 4200 do
        begin
        na_file[k]:=0
        end;

  assign(sein,'c:\dsp_werk\ontvang.prn');
  reset(sein);
  l:=1;
  r:=1;
for   k:=1 to 4000 do
begin
    readln(sein,van_file[k]);
end;
  TextColor(Red+Blink);
  TextBackground(black);
```

```
writeln('Sender met viterbi');

BaseS:=$2c0;
BaseR:=$2c0;
timeout:=10000;
BaseAddress:=$2b0;

DSP_Reset(BaseS);
DSP_Load(BaseS,'c:\dsp_werk\2c0\senderv');
DSP_Reset(BaseAddress);

DSP_load($2b0,'c:\dsp_werk\reset');
DSP_Go($2b0);

DSP_Reset(BaseAddress);
DSP_load(baseaddress,'c:\dsp_werk\ontvang\ontvangv');
DSP_Go(BaseS);

for   k:=1 to 1000 do

begin
                DSP_WriteFlag(BaseS,timeout);
                DSP_WriteLongint(BaseS,van_file[k]);
end;

fout:=0;

DSP_Go(BaseAddress);

 DP_ReadFlag($2b0,timeout);
{ fout:=0;}

  while  (fout=1) do
  begin;
  DSP_Reset(BaseAddress);
  DSP_load(baseaddress,'c:\dsp_werk\ontvang\ontvangv');
  DSP_Reset(BaseS);
  DSP_Load(BaseS,'c:\dsp_werk\2c0\senderv');
   DSP_Go(BaseS);

      for   k:=1 to 1000 do
         begin
              DSP_WriteFlag(BaseS,timeout);
              DSP_WriteLongint(BaseS,van_file[k]);
         end;
  DSP_Go(BaseAddress);
  DP_ReadFlag(BaseAddress,timeout);
  end;

for i:=1 to 2 do
```

```
 begin
  TextBackground(white);
   delay(10000);
  end;

 for i:=1 to 4000 do
 begin
   DP_ReadFlag(BaseAddress,timeout);
   DSP_ReadLongInt(BaseAddress,data[i]);
 end;

 assign(datafile1,'c:\dsp_werk\sinus.prn');
 rewrite(datafile1);

 assign(datafile2,'c:\dsp_werk\data.prn');
 rewrite(datafile2);

 assign(datafile3,'c:\dsp_werk\cosinus.prn');
 rewrite(datafile3);

{  for   qq:=1 to 2000 do
 begin;

 for i:=1 to 8 do
 begin
   writeln(datafile3,data[i]);
 end;
 end;

for   qq:=1 to  2000 do
 begin;

 for i:=9 to 16 do
 begin
   writeln(datafile1,data[i]);
 end;
 end;       }

 for i:=32 to 4000 do
 begin
   writeln(datafile2,data[i]);
 end;
 close(datafile1);
 close(datafile3);
 close(datafile2);
 close(sein);
 writeln('Data geskryf na ontvang.DAT');
 end.
end.
end.
```

Bylae D: Sender DSP Sagteware.

```
;Toekenning van adresse aan registers.

bcr             equ     $fffe
ipr             equ     $ffff
pcc             equ     $ffe1
pcddr           equ     $ffe3
pcd             equ     $ffe5
ssr             equ     $ffee
cra             equ     $ffec
crb             equ     $ffed
srx             equ     $ffef
stx             equ     $ffef
in_1            equ     $100
in_2            equ     $101
in_w            equ     $102
in_w1           equ     $103
skuifreg1       equ     $211
skuifreg2       equ     $212
bit1            equ     $213
bit2            equ     $214
bit3            equ     $215
sin_end         equ     $208
sin_beg         equ     $200
vier_vlakI      equ     $216
vier_vlakQ      equ     $217
skuif_w         equ     $218

        opt     fc,mu,s,w,mex
        org     p:$0000
        jmp     Main
        org     p:$100

Main
        jsr     sin_draer       ;Die sinustabel van draer word opgestel.
        jsr     sin_fases       ;Die sinustabel van die gemoduleerde sein word opgestel.
        jsr     Setup           ;Die digitaal-na-analoog omskakelaar word opgestel.
        jsr     reg2_reg4
        do      #1000,innnn
        jsr     inw             ;Datawoord word ingelees.
        move    x:in_w1,a1
        move    a1,x:(r4)+n4    ;Plaas woord in regte adres.

innnn
        move    #$400,r4
        move    #$300,r2
        do      #1600,fff
        nop
        nop
        nop

fff
        do      #800,fff1
        nop
```

```
        nop
        nop

fff1
        do          #1600,fff2
        nop
        nop
        nop

fff2
        do          #1600,fff3
        nop
        nop
        nop

fff3

qq1
        do          #1600,draer         ;Versend 1600/8=200 draaggolf siklusse.
        jsr         output

draer
        move        #$200,r2
        do          #40,sinch           ;Versend sinchroniseersein.
        move        #4,a1
        move        a1,x:skuif_w
        jsr         reg_pos
        do          #8,uiteee
        jsr         output

uiteee
        move    #0,a1
        move    a1,x:skuif_w
        jsr     reg_pos
        do      #8,uiteee1
        jsr     output

uiteee1
        nop

sinch
        move    #$400,r4

nul
        move    #$200,r2
        move    #0,a1
        move    a1,x:skuif_w
        do      #6,uite
        jsr     reg_pos
        do      #8,uitee
        jsr     output

uitee
        nop

uite
bo
        move    #$400,r4
        move    #$200,r2
        do      #1000,uit12345
```

```
        nop
        move    x:(r4)+n4,a1
        move    a1,x:in_w
        do      #8,uit1234
        jsr     woord
        jsr     skuif                   ;Plaas datawoord in skuifregisters.
        jsr     enkodeerder             ;Doen enkodering.
        jsr     skuif1
        jsr     reg_pos
        do      #8,uit123               ;Lees 8 monsters uit.
        jsr     output

uit123
        nop
        nop

uit1234
        nop

uit12345
        jmp     bo
end

reg2_reg4
        move    #$400,r4
        move    #1000,m4
        move    #1,n4
        move    #$200,r2
        move    #7,m2
        move    #1,n2
        rts

woord
        clr   a
        move            a,x:in_1
        move            a,x:in_2
        btst            #0,x:in_w
        jscs            een
        btst            #1,x:in_w
        jscs            twee
        move            x:in_w,a1
        lsr             a
        lsr             a
        move            a1,x:in_w
        rts

een
        move            #>$1,a1
        move            a1,x:in_1
        rts

twee
        move            #>$1,a1
        move            a1,x:in_2
        rts
;*************************************************************
;Die digitaal-na-analoog omskakelaar word opgestel.
;*************************************************************
Setup
        clr             b
```

```
        move    b1,x:$FFEF
        move    #%0100000000000000,a
        move    a,x:$FFEC
        move    #%0011101000000000,a
        move    a,x:$FFED

        move    #%0000000111111111,a
        move    a,x:$FFE1
        move    #%0000000100001110,a
        move    a,x:$FFE3
        move    #%0000000000000000,a
        move    a,x:$FFFF

freq
        movep   #$ffff1e,y:$ffe0
        clr             b
        move    #$ffffff,a1
        move    #$2c5800,b1
        btst            #6,x:$ffee
        jcc             pri3
        move    a1,x:$ffef

sec3
        btst            #6,x:$ffee
        jcc             sec3
        move    b1,x:$ffef
         move           #$489200,b1

pri31
        btst            #6,x:$ffee
        jcc             pri31
        move    a1,x:$ffef


sec31
        btst            #6,x:$ffee
        jcc             sec31
        move    b1,x:$ffef
        clr             b
        rts
        move    #$00e700,b1

pri311
        btst            #6,x:$ffee
        jcc             pri311
        move    a1,x:$ffef

sec311
        btst            #6,x:$ffee
        jcc             sec311
        move    b1,x:$ffef
        clr             b
        rts

;************************************************
;Uitset
;************************************************

output
uit1
```

```
        move    #>$FFFc00,b1
        move    b1,x0

uit2
        btst            #6,x:$FFEE
        jcc             uit2
        move    x:(r2)+,b1
        and     x0,b
        move    b1,x:$FFEF
        rts

;*************************************************
; Bepaling van register posisie
;*************************************************


reg_pos
        move            #$200,r2
        move            x:skuif_w,a1
        move            a1,n2
        nop
        move            (r2)+n2
        move            #>1,n2
        rts

;*************************************************
; Aantal skuiwe op sinustabel word bepaal.
;*************************************************

skuif1
        move            #>0,x1
        move            x1,x:skuif_w
        .IF     x:bit3 <EQ> #>0 and x:bit2 <EQ> #>0 and x:bit1 <EQ> #>0
        move    #>0,a1
        move    a1,x:skuif_w
        .ENDI
        .IF     x:bit3 <EQ> #>0 and x:bit2 <EQ> #>0 and x:bit1 <EQ> #>1
         move   #>1,a1
         move   a1,x:skuif_w
        .ENDI
        .IF     x:bit3 <EQ> #>0 and x:bit2 <EQ> #>1 and x:bit1 <EQ> #>0
         move   #>2,a1
         move   a1,x:skuif_w
        .ENDI
        .IF     x:bit3 <EQ> #>0 and x:bit2 <EQ> #>1 and x:bit1 <EQ> #>1
         move   #>3,a1
         move   a1,x:skuif_w
        .ENDI
        .IF     x:bit3 <EQ> #>1 and x:bit2 <EQ> #>0 and x:bit1 <EQ> #>0
         move   #>4,a1
         move   a1,x:skuif_w
        .ENDI
        .IF     x:bit3 <EQ> #>1 and x:bit2 <EQ> #>0 and x:bit1 <EQ> #>1
         move   #>5,a1
         move   a1,x:skuif_w
        .ENDI
        .IF     x:bit3 <EQ> #>1 and x:bit2 <EQ> #>1 and x:bit1 <EQ> #>0
         move   #>6,a1
         move   a1,x:skuif_w
        .ENDI
```

```
        .IF     x:bit3 <EQ> #>1 and x:bit2 <EQ> #>1 and x:bit1 <EQ> #>1
         move   #>7,a1
         move   a1,x:skuif_w
        .ENDI

;************************************************
;
; Sinustabel word opgestel.
;************************************************
;

sin_draer
        move    #0,r7
        move    #1,n7
        move    #7,m7
        move    #0,x0
        move    x0,x:bit1
        move    x0,x:bit2
        move    x0,x:bit3
        move    x0,x:in_1
        move    x0,x:in_2
        move    x0,x:skuifreg1
        move    x0,x:skuifreg2
        move    #>$0,x0
        move    x0,X:$300
        move    #$2c3000,x0
        move    x0,x:$301
        move    #$3e8000,x0
        move    x0,x:$302
        move    #$2c3000,x0
        move    x0,x:$303
        move    #$0,x0
        move    x0,x:$304
        move    #-$2c3000,X0
        move    x0,x:$305
        move    #-$3e8000,x0
        move    x0,x:$306
        move    #-$2c3000,x0
        move    x0,x:$307
        move    #sin_beg,r1
        move    #sin_beg,r2
        rts
;************************************************************
;
;Sinustabel van gemoduleerde sein word opgestel.
;************************************************************
;

sin_fases
        move    #>$17f000,x0
        move    x0,X:$200
        move    #>$3e9000,x0
        move    x0,x:$201
        move    #$3e9000,x0
        move    x0,x:$202
        move    #$17f000,x0
        move    x0,x:$203
        move    #-$17f000,x0
        move    x0,x:$204
        move    #-$3e9000,X0
        move    x0,x:$205
        move    #-$3e9000,x0
        move    x0,x:$206
        move    #-$17f000,x0
```

```
        move    x0,x:$207
        rts

inw
        btst            #0,x:$ffe9
        jcc             inw
        move    x:$ffeb,a
        move    a1,x:in_w1
        rts

;*****************************
;Enkodering word gedoen.
;*****************************

enkodeerder
        clr     a
        move            #>0,a1
        move            a1,x:bit1
        move            #>2,a1
        move            x:skuifreg2,x1
        and             x1,a1
        .IF             a1  <EQ> #>2
        move            #>1,a1
        move            a1,x:bit1
        .ENDI
        clr     a
        move            #>0,y1
        move            #>0,a1
        move            a1,x:bit2
        move            #>4,a1
        move            x:skuifreg2,x1
        and             x1,a1
        .IF             a1  <EQ> #>4
        move            #>1,y1
        .ENDI
        clr             a
        move            #>0,x1
        move            #>1,a1
        move            x:skuifreg1,x1
        and             x1,a1
        .IF             a1  <EQ> #>1
        move            #>1,a1
        .ENDI
        eor             y1,a1
        move            a1,x:bit2
        move            #>0,y1
        clr     a
        move            #>0,a1
        move            a1,x:bit3
        move            #>1,a1
        move            x:skuifreg2,x1
        and             x1,a1
        .IF             a1  <EQ> #>1
        move            #>1,y1
        .ENDI
        move            #>0,x1
        clr             a
        move            #>2,a1
        move            x:skuifreg1,x1
        and             x1,a1
```

```
        .IF             a1  <EQ> #>2
        move            #>1,a1
        .ENDI
        eor             y1,a1
        move            a1,x:bit3
        rts

skuif
        clr             a
        move            x:skuifreg1,a1
        lsl     a
        move            a1,x:skuifreg1
        clr     a
        .IF             x:in_1  <EQ> #>1
        move            x:skuifreg1,a1
        bset            #0,a1
        move            a1,x:skuifreg1
        .ENDI
        clr     a
        move    x:skuifreg2,a1
        lsl     a
        move    a1,x:skuifreg2
        clr     a
        .IF     x:in_2  <EQ> #>1
        move    x:skuifreg2,a1
        bset    #0,a1
        move    a1,x:skuifreg2
        .ENDI
        rts
```

Bylae E: Ontvanger DSP Sagteware.

```
;***********************************************
;Stel registers op.
;***********************************************

ROUND           equ     0
posk            equ     6000
poske           equ     6
pos1            equ     1250
posg            equ     14000
posge           equ     14
nul1            equ     0
negk            equ     -6000
negke           equ     -6
neg1            equ     -1250
negg            equ     -14000
negge           equ     -14
uit_waarde      equ     $201
in_waarde       equ     $202
sin_ant         equ     $203
cos_ant         equ     $204
sin_filt        equ     $205
cos_filt        equ     $206
QIC             equ     $207
Feerste         equ     $208
Ftweede         equ     $209
Fantwoord       equ     $20a
Feerste1        equ     $20b
som_ant         equ     $20c
ant_v           equ     $36c
ant             equ     $20d
ant_0           equ     $20e
dek_ant         equ     $20f
regs0           equ     $216
regs            equ     $210
regs2           equ     $211
regs3           equ     $215
Fantw           equ     $212
ant_uit         equ     $213
klein           equ     $217
ont             equ     $214
Fant0           equ     $300
Fant1           equ     $301
Fant2           equ     $302
Fant3           equ     $303
Fant4           equ     $304
Fant5           equ     $305
Fant6           equ     $306
Fant7           equ     $307

        opt             fc,mu,s,w,mex

begin
        equ     $0080
```

```
        org             p:$0000
        jmp             _begin
        clr             a
        move            a,x:$fffe
        org             p:begin


;************************************************
;Maak seker x geheu is skoon.
;************************************************
_begin
        jsr             opstel_v

;************************************************
; Stel digitaal na analoog omskakelaar op.
;************************************************
opstel_AIC1
        jsr             setup
        do              #10,non
        jsr     in

non
        move    #0,r7
        move    #1,n7
        move    #15,m7
        jsr             reg2_reg3
;************************************************
; Stel sinus en kosinustabelle op.
;************************************************

jsr             sin_tabel1
        .IF     x:$220  <GT>  #>20  OR x:$220 <LT>  #>5
        jmp     opstel_AIC1
        .ENDI
        do      #4,www
        .IF     x:(r2)+  <LT>   #>0
        jmp             opstel_AIC1
        .ENDI
        nop

www
        do              #4,www1
        .IF             x:(r2)+  <GT>   #>0
        jmp             opstel_AIC1
        .ENDI
        nop

www1
        jsr             reg2_reg3
        jsr             cos_tabel1
        clr             a
        move            #$600,r1
        clr             a
        move            a1,r1

;************************************************
; Sinchroniseer met sender.
;************************************************

DDDDD
```

```
        jsr             SSSSS1
        .IF             x:ant  <EQ>  #>4
        move            (r1)+
        .ENDI
        .IF             x:ant  <NE>  #>4
        clr             a
        move            a1,r1
        jmp             DDDDD
        .ENDI
        jsr             SSSSS1
        .IF     x:ant   <EQ>  #>0
        move            (r1)+
        .ENDI
        .IF     x:ant   <NE>  #>0
        clr             a
        move            a1,r1
        .ENDI
        .If             r1  <EQ>   #>80
        jmp             SSSSS
        .ENDI
        jmp             DDDDD

SSSSS
        clr             a
        move            a1,x:uit_waarde
        jsr             uitt
        move            #$600,r1
        jmp             uuit
        do              #1600,zzz1
        clr             a
        move            a1,x:sin_ant
        move            a1,x:cos_ant
        move            a1,x:dek_ant
        move            a1,x:in_waarde
        do              #8,hhh
        jsr             in1
        jsr             sind
        jsr             cosd

hh
        jsr             deel4_s
        jsr             deel4_c
        clr             a

zzz1
    jmp                 q44


uuit

;*****************************************************
;Doen demodulasie.
;*****************************************************

beg_
        move            #$600,r1
        .LOOP           #5
        jsr             beg_hier
        .ENDL
        jmp             w44
```

```
beg_hier
        do      #1000,q44
        do      #8,q3
        clr     a
        move    a1,x:sin_ant
        move    a1,x:cos_ant
        jsr     in1             ;Lees monster in vanaf analoog-na-digitale omskakelaar.
        jsr     sind            ;Vermenigvuldig ingeleesde waarde met sinus.
        jsr     cosd            ;Vermenigvuldig ingeleesde waarde met  kosinus.
        jsr     PUNT_A          ;Bepaal moontlike afstande na NODUS-A
        jsr     PUNT_B          ;Bepaal moontlike afstande na NODUS-B
        jsr     in1             ;Lees monster in vanaf analoog-na-digitale omskakelaar.
        jsr     sind            ;Vermenigvuldig ingeleesde waarde met sinus.
        jsr     cosd            ;Vermenigvuldig ingeleesde waarde met  kosinus.
        jsr     PUNT_C          ;Bepaal moontlike afstande na NODUS-C
        jsr     PUNT_D          ;Bepaal moontlike afstande na NODUS-D
        jsr     in1             ;Lees monster in vanaf analoog-na-digitale omskakelaar.
        jsr     sind            ;Vermenigvuldig ingeleesde waarde met sinus.
        jsr     cosd            ;Vermenigvuldig ingeleesde waarde met  kosinus.
        jsr     PUNT_E          ;Bepaal moontlike afstande na NODUS-E
        jsr     PUNT_F          ;Bepaal moontlike afstande na NODUS-F
        jsr     in1             ;Lees monster in vanaf analoog-na-digitale omskakelaar.
        jsr     sind            ;Vermenigvuldig ingeleesde waarde met sinus.
        jsr     cosd            ;Vermenigvuldig ingeleesde waarde met  kosinus.
        jsr     PUNT_G          ;Bepaal moontlike afstande na NODUS-G
        jsr     PUNT_H          ;Bepaal moontlike afstande na NODUS-H
        jsr     in1             ;Lees monster in vanaf analoog-na-digitale omskakelaar.
        jsr     sind            ;Vermenigvuldig ingeleesde waarde met sinus.
        jsr     cosd            ;Vermenigvuldig ingeleesde waarde met  kosinus.
        jsr     in1             ;Lees monster in vanaf analoog-na-digitale omskakelaar.
        jsr     sind            ;Vermenigvuldig ingeleesde waarde met sinus.
        jsr     cosd            ;Vermenigvuldig ingeleesde waarde met  cosinus.
        move    r1,a1
        move    a1,x:regs
        move    r2,a1
        move    a1,x:regs2
        move    x:regs2,a1
        move    a1,r2
        move    x:regs,a1
        move    a1,r1
        move    r1,a1
        move    a1,x:regs
        move    x:regs,a1
        move    a1,r1
        jsr     in1             ;Lees monster in vanaf analoog-na-digitale omskakelaar
        jsr     sind            ;Vermenigvuldig ingeleesde waarde met sinus.
        jsr     cosd            ;Vermenigvuldig ingeleesde waarde met  cosinus.
        jsr     in              ;Lees monster in vanaf analoog-na-digitale omskakelaar
        jsr     sind            ;Vermenigvuldig ingeleesde waarde met sinus.
        jsr     cosd            ;Vermenigvuldig ingeleesde waarde met  cosinus.
        jsr     deel4_s         ;Doen filtreering op infase kanaal.
        jsr     deel4_c         ;Doen filtreering op kwadratuurfase kanaal.

        move    r1,a1
        move    a1,x:regs
        move    x:regs,a1
        move    a1,r1
        clr     a
        move    a1,x:sin_filt
```

```
        move            a1,x:cos_filt

sin_cos
        clr             a
        move    a1,x:ant
        jsr     dekodeerder
        move    r1,a1
        move    a1,x:regs
        move    r3,a1
        move    a1,x:regs3
        move    r0,a1
        move    a1,x:regs0
        move    x:regs3,a1
        move    a1,r3
        move    x:regs,a1
        move    a1,r1
        move    x:regs0,a1
        move    a1,r0
        jsr     dekodeer2
        move    (r7)+
        move    x:ant,a1
        move    a1,x:ant_0
        jsr     skuif_int

q3
        clr     b
        move    x:ant_uit,a1
        do      #16,aaa1
        asl     b
        btst    #0,a1
        jscs    sett
        asr     a

aaa1
        nop
        move    b1,x:ant_uit
        move    b1,x:(r1)+

q44
    rts

w44
        jsr     reg2_reg3
        do      #8,q45
        move    x:(r3)+,a1
        move    a1,x:uit_waarde
        jsr     uitt                    ;Ontvangde data word na rekenaar gestuur.

q46
        move    #$600,r1
        do      #4000,q4
        move    x:(r1)+,a1
        move    a1,x:uit_waarde
        jsr     uitt

q4
        jmp     beg_
```

```
SSSSS1
        clr             a
        move    a1,x:sin_ant
        move    a1,x:cos_ant
        move    a1,x:dek_ant
        move    a1,x:in_waarde
        do      #8,hhh11
        jsr     in1
        jsr     sind
        jsr     cosd

hhh11
        jsr     deel4_s
        jsr     deel4_c
        clr     a
        move    a1,x:ant
        jsr     dekodeerder
        rts

;*********************************************************
;Die infase kanaal word gefiltreer.
;x:sin_filt=x:sin_filt/4
;*********************************************************
deel4_s
        clr             a
        move            x:sin_ant,a
        tst             a
        jpl             pos
        abs             a
        asr             a
        asr             a
        asr             a
        neg             a
        move            a,x:sin_ant
        move            a,x:sin_filt
        rts

pos
        asr             a
        asr             a
        asr             a
        move            a,x:sin_ant
        move            a,x:sin_filt    ;
        rts
;*********************************************************
;Die kwadratuurfase kanaal word gefiltreer.
;x:cos_filt=x:cos_filt/4
;*********************************************************

deel4_c
        clr             a
        move            x:cos_ant,a
        tst             a
        jpl             posc
        abs             a
        asr             a
        asr             a
        asr             a
        neg             a
        move            a,x:cos_ant
```

```
        move            a,x:cos_filt
        rts

posc
        asr             a
        asr             a
        asr             a
        move            a,x:cos_ant
        move            a,x:cos_filt
        rts

deel6_c
        move            x:cos_ant,a
        move            #>6,x0
        tfr             a,b
        abs             a
        clr     a       a1,x1
        move            x1,a0
        asl             a
        rep             #$18
        div             x0,a
        eor             x0,b
        jpl             L2
        neg             a

L2
        move            a0,a
        move            a1,x:cos_ant
        rts

sett
        bset    #0,b1
        rts

;*********************************************************************
;Dekodeerder.
;*********************************************************************

skuif_int
        clr             a
        move            x:ant_uit,a1
        btst            #0,x:dek_ant
        ROL             a
        btst            #1,x:dek_ant
        ROL             a
        move            a1,x:ant_uit
        rts
        .IF             x:sin_ant <GT> #pos1
        jsr             cos1
        rts
        .ENDI
        .IF             x:sin_ant <LT> #neg1
        jsr             cos2
        rts
        .ENDI
        .IF             x:sin_ant <GT> #0 and x:sin_ant <LT> #pos1
        jsr             cos3
        rts
        .ENDI
        .IF             x:sin_ant <LT> #0 and x:sin_ant <GT> #neg1
```

```
        jsr         cos4
        rts
        .ENDI
        rts

cos1
        .IF         x:cos_ant <LT> #0 and x:cos_ant <GT> #neg1
        move        #>7,a1
        move        a1,x:ant
        rts
        .ENDI
        .IF         x:cos_ant <GT> #0 and x:cos_ant <LT> #pos1
        move        #>0,a1
        move        a1,x:ant
        rts
        .ENDI
        rts

cos2
        .IF         x:cos_ant <GT> #0 and x:cos_ant <lt> #pos1
        move        #>3,a1
        move        a1,x:ant
        rts
        .ENDI
        .IF         x:cos_ant <LT> #0 and x:cos_ant <GT> #neg1
        move        #>4,a1
        move        a1,x:ant
        rts
        .ENDI
        rts

cos3
        .IF         x:cos_ant  <GT> #pos1
        move        #>1,a1
        move        a1,x:ant
        rts
        .ENDI
        .IF         x:cos_ant <LT> #neg1
        move        #>6,a1
        move        a1,x:ant
        .ENDI
        rts

cos4
        .IF         x:cos_ant <LT> #neg1
        move        #>5,a1
        move        a1,x:ant
        rts
        .ENDI
        .IF         x:cos_ant <GT> #pos1
        move        #>2,a1
        move        a1,x:ant
        rts
        .ENDI
        rts
```

```
;*****************************************************************
;
;Kosinustabel.
;*****************************************************************
;

cos_tabel1
        move    x:$222,a1
        move    a1,x:$230
        move    x:$223,a1
        move    a1,x:$231
        move    x:$224,a1
        move    a1,x:$232
        move    x:$225,a1
        move    a1,x:$233
        move    x:$226,a1
        move    a1,x:$234
        move    x:$227,a1
        move    a1,x:$235
        move    x:$220,a1
        move    a1,x:$236
        move    x:$221,a1
        move    a1,x:$237
        rts

;*****************************************************************
;
;Opstel van registers 2 en 3
;*****************************************************************
;


reg2_reg3
        move    #>$220,r2
        move    #>1,n2
        move    #>7,m2
        move    #>$230,r3
        move    #>1,n3
        move    #>7,m3
        rts

q2
        clr     a
        do      #8,hada
        move    a1,x:(r2)+n2

hada
        jsr     in
        .IF     x:in_waarde <GT>  #>0
        jmp     hada
        .ENDI

q222
        jsr     in
        .IF     x:in_waarde <LT>  #>0
        jmp     q222
        .ENDI
        do      #7,ha1
        jsr     in
ha1
        do      #8,hahaha
        jsr     in
```

```
        move    x:in_waarde,a1
        move    a1,x:(r2)+n2

hahaha
        rts

sin_tabel2
          clr           a
          do            #8,q111
          move          a1,x:(r2)+n2

q111
        move            #$220,r2
        do              #8,q11
        jsr             in1
        move            x:in_waarde,a1
        move            a1,x:(r2)+n2

q11
        rts

;*********************************************************************
;Inlees van monsters.
;*********************************************************************

in
        clr             a
        move            a1,x:in_waarde

inn
        btst            #6,x:$ffee
        jcc             inn
        movep           x:$ffef,b
        move            #$fffc00,x0
        and             x0,b
        do              #15,inns
        asr             b
inns
        move            b,X:in_waarde
        move            #0,b
        movep           b,X:$ffef
        rts

in1
        clr             a
        move            a1,x:in_waarde

inn1
        btst            #6,x:$ffee
        jcc             inn1
        movep           x:$ffef,b
        move            #$fffc00,x0
        and             x0,b
        do              #15,inns1
        asr             b
inns1
        move            b,X:in_waarde
        move            #0,b
        movep           b,X:$ffef
        rts
```

```
;*********************************************************************
Uitlees van data na rekenaar.
;*********************************************************************
uitt

pcst
        btst        #1,x:$ffe9
        jcc         pcst
        nop
        move        x:uit_waarde,a1
        movep       a1,x:$ffeb
        rts

;*********************************************************************
;Monster word met sinus vermenugvuldig.
;Sin-ant=in-Waarde*sin
;*********************************************************************
sind

        clr         a
        move        a,x0
        move        a,y0
        move        a,x1
        move        x:in_waarde,x0
        move        x:(r2)+n2,y0
        mpy         y0,x0,a
        asr         a
        move        a0,a
        move        x:sin_ant,x1
        add         x1,a
        move        a1,x:sin_ant
        rts


;*********************************************************************
;Monster word met kosinus vermenigvuldig.
;cos_ant=in_waarde*cos
;*********************************************************************

cosd
        clr     a
        move    a,x0
        move    a,y0
        move    a,x1
        move    x:in_waarde,x0
        move    x:(r3)+,y0
        mpy     y0,x0,a
        asr     a
        move    a0,a
        move    x:cos_ant,x1
        add     x1,a
        move    a1,x:cos_ant
        rts

reg3_sin
        move    #$100,r3
        move    x:Fantwoord,x1
        move    x1,n3
        nop
        move    (r3)+n3
```

```
        nop
        move    #32,n3
        move    #$ff,m3
        rts

waardes
din1
        btst    #6,x:$ffee
        jcc     din1
        move    x:$ffef,b
        move    #$fffc00,x0
        and     x0,b
        move    b,X:Feerste
        move    #0,b
        move    b,X:$ffef
        .IF     x:Feerste <GT> #0
        jmp     din1
        .ENDI

din2
        btst    #6,x:$ffee
        jcc     din2
        move    x:$ffef,b
        move    #$fffc00,x0
        and     x0,b
        move    b,X:Ftweede
        move    #0,b
        move    b,X:$ffef
        .IF     X:Ftweede <LT> #0
        jmp     din2
        .ENDI
        rts

;*********************************************************************
;Digitaal-na analoog omskakelaar word opgestel.
;*********************************************************************

setup
        ori     #%00000100,omr
        clr     b
        move    b1,x:$FFEF
        move    #$4000,a
        move    a,x:$FFEC
        move    #$3A00,a
        move    a,x:$FFED
        movep           #$FFFF1E,y:$FFE0
        move    #$01FF,a
        move    a,x:$FFE1
        move    #$010E,a
        move    a,x:$FFE3
        move    #$0000,a
        move    a,x:$FFFF
        move    #>0,r0
        move    #$FFFC00,b1
        move    b1,x0
        clr     a
        clr     b
        move    a1,x:$ffef

pri1
```

```
        btst    #6,x:$ffee
        jcc     pri1
        clr     b
        move    #$ffffff,b
        move    b,x:$ffef

sec1
        btst    #6,x:$ffee
        jcc     sec1
        clr     b
        move    #$2c5800,b
        move    b,x:$ffef
        clr     b

pri2
        btst    #6,x:$ffee
        jcc     pri2
        clr     b
        move    #$ffffff,b
        move    b,x:$ffef

sec2
        btst    #6,x:$ffee
        jcc     sec2
        clr     b
        move    #$489200,b
        move    b,x:$ffef
        clr     b

pri222
        btst    #6,x:$ffee
        jcc     pri222
        clr     b
        move    #$ffffff,b
        move    b,x:$ffef

sec222
        btst    #6,x:$ffee
        jcc     sec222
        clr     b
        move    #$00e700,b
        move    b,x:$ffef
        clr     b
        do      #256,capt1

datai
        btst    #6,x:$ffee
        jcc     datai
        nop

datao
        and     x0,b
        movep   b,x:$FFEF
        move    #0,b
capt1

        rts
```

```
;*************************************************************************
;Bepaling van kortste pad na nodus sowel as afkomstige nodus.
;*************************************************************************

PUNT_A
        move        r1,a1
        move        a1,x:regs
        move        r2,a1
        move        a1,x:regs2
        move        y:$300,x0
        move        x0,x:$321
        move        y:$304,x0
        move        x0,x:$322
        move        y:$302,x0
        move        x0,x:$323
        move        y:$306,x0
        move        x0,x:$324
        move        #$310,r1
        jsr         AFSTANDE            ;Die stoor van afstande vanaf vorige punte.
        move        #$340,r2
        jsr         KORTSTE_STAP        ;Bepaling van kortste stap vanaf vorige punte.
        move        #$310,r1
        move        #$400,r6
        move        #$10,n6
        jsr         SKRYF_IN_MATRIKS    ;Skryf van wenner sowel as afkomstige pad in matriks.
        move        x:regs,a1
        move        a1,r1
        move        x:regs2,a1
        move        a1,r2
        rts

;*************************************************************************
;Bepaling van kortste pad na nodus sowel as afkomstige nodus.
;*************************************************************************

PUNT_B
        move    r1,a1
        move    a1,x:regs
        move    r2,a1
        move    a1,x:regs2
        move    y:$304,x0
        move    x0,x:$321
        move    y:$300,x0
        move    x0,x:$322
        move    y:$306,x0
        move    x0,x:$323
        move    y:$302,x0
        move    x0,x:$324
        move    #$310,r1
        jsr     AFSTANDE            ;Die stoor van afstande vanaf vorige punte.
        move    #$340,r2
        jsr     KORTSTE_STAP        ;Bepaling van kortste stap vanaf vorige punte.
        move    #$314,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS    ;Skryf van wenner sowel as afkomstige pad in matriks.
        move    x:regs,a1
        move    a1,r1
        move    x:regs2,a1
        move    a1,r2
        rts
```

```
;*************************************************************************
;Bepaling van kortste pad na nodus sowel as afkomstige nodus.
;*************************************************************************

PUNT_C
        move    r1,a1
        move    a1,x:regs
        move    r2,a1
        move    a1,x:regs2
        move    y:$302,x0
        move    x0,x:$321
        move    y:$306,x0
        move    x0,x:$322
        move    y:$300,x0
        move    x0,x:$323
        move    y:$304,x0
        move    x0,x:$324
        move    #$310,r1
        jsr     AFSTANDE            ;Die stoor van afstande vanaf vorige punte.
        move    #$340,r2
        jsr
        KORTSTE_STAP                ;Bepaling van kortste stap vanaf vorige punte.
        move    #$311,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS    ;Skryf van wenner sowel as afkomstige pad in matriks.
        move    x:regs,a1
        move    a1,r1
        move    x:regs2,a1
        move    a1,r2
        rts
;*************************************************************************
;Bepaling van kortste pad na nodus sowel as afkomstige nodus.
;*************************************************************************
PUNT_D
        move    r1,a1
        move    a1,x:regs
        move    r2,a1
        move    a1,x:regs2
        move    y:$306,x0
        move    x0,x:$321
        move    y:$302,x0
        move    x0,x:$322
        move    y:$304,x0
        move    x0,x:$323
        move    y:$300,x0
        move    x0,x:$324
        move    #$310,r1
        jsr     AFSTANDE            ;Die stoor van afstande vanaf vorige punte.
        move    #$340,r2
        jsr     KORTSTE_STAP        ;Bepaling van kortste stap vanaf vorige punte.
        move    #$315,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS    ;Skryf van wenner sowel as afkomstige pad in matriks.
        move    x:regs,a1
        move    a1,r1
        move    x:regs2,a1
        move    a1,r2
    rts

;*************************************************************************
```

```
;Bepaling van kortste pad na nodus sowel as afkomstige nodus.
;*************************************************************************

PUNT_E
        move    r1,a1
        move    a1,x:regs
        move    r2,a1
        move    a1,x:regs2
        move    y:$301,x0
        move    x0,x:$321
        move    y:$305,x0
        move    x0,x:$322
        move    y:$303,x0
        move    x0,x:$323
        move    y:$307,x0
        move    x0,x:$324
        move    #$314,r1
        jsr     AFSTANDE            ;Die stoor van afstande vanaf vorige punte.
        move    #$344,r2
        jsr     KORTSTE_STAP        ;Bepaling van kortste stap vanaf vorige punte.
        move    #$312,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS    ;Skryf van wenner sowel as afkomstige pad in matriks.
        move    x:regs,a1
        move    a1,r1
        move    x:regs2,a1
        move    a1,r2
        rts

;*************************************************************************
;Bepaling van kortste pad na nodus sowel as afkomstige nodus.
;*************************************************************************

PUNT_F
        move    r1,a1
        move    a1,x:regs
        move    r2,a1
        move    a1,x:regs2
        move    y:$305,x0
        move    x0,x:$321
        move    y:$301,x0
        move    x0,x:$322
        move    y:$307,x0
        move    x0,x:$323
        move    y:$303,x0
        move    x0,x:$324
        move    #$314,r1
        jsr     AFSTANDE            ;Die stoor van afstande vanaf vorige punte.
        move    #$344,r2
        jsr     KORTSTE_STAP        ;Bepaling van kortste stap vanaf vorige punte.
        move    #$316,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS    ;Skryf van wenner sowel as afkomstige pad in matriks.
        move    x:regs,a1
        move    a1,r1
        move    x:regs2,a1
        move    a1,r2
        rts
;*************************************************************************
;Bepaling van kortste pad na nodus sowel as afkomstige nodus.
```

```
;*************************************************************************

PUNT_G
        move    r1,a1
        move    a1,x:regs
        move    r2,a1
        move    a1,x:regs2
        move    y:$303,x0
        move    x0,x:$321
        move    y:$307,x0
        move    x0,x:$322
        move    y:$301,x0
        move    x0,x:$323
        move    y:$305,x0
        move    x0,x:$324
        move    #$314,r1
        jsr     AFSTANDE            ;Die stoor van afstande vanaf vorige punte.
        move    #$344,r2
        jsr     KORTSTE_STAP        ;Bepaling van kortste stap vanaf vorige punte.
        move    #$313,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS    ;Skryf van wenner sowel as afkomstige pad in matriks.
        move    x:regs,a1
        move    a1,r1
        move    x:regs2,a1
        move    a1,r2
        rts
;*************************************************************************
;Bepaling van kortste pad na nodus sowel as afkomstige nodus.
;*************************************************************************

PUNT_H
        move    r1,a1
        move    a1,x:regs
        move    r2,a1
        move    a1,x:regs2
        move    y:$307,x0
        move    x0,x:$321
        move    y:$303,x0
        move    x0,x:$322
        move    y:$305,x0
        move    x0,x:$323
        move    y:$301,x0
        move    x0,x:$324
        move    #$314,r1
        jsr     AFSTANDE            ;Die stoor van afstande vanaf vorige punte.
        move    #$344,r2
        jsr     KORTSTE_STAP        ;Bepaling van kortste stap vanaf vorige punte.
        move    #$317,r1
        lua     (r6)+n6,r6
        jsr     SKRYF_IN_MATRIKS    ;Skryf van wenner sowel as afkomstige pad in matriks.
        move    x:regs,a1
        move    a1,r1
        move    x:regs2,a1
        move    a1,r2
        rts

AFSTANDE                            ;Die stoor van afstande vanaf vorige punte.
        move    #>$321,r2
        .LOOP   #$4
```

```
        move    x:(r1),a
        move    x:(r2),x0
        add     x0,a
        move    a,y:(r2)
        move    (r1)+
        move    (r2)+
        .ENDL
        rts

KORTSTE_STAP
        move    #>$321,r1
        move    #>$1,n1
        move    #>$1,n2
        move    y:(r1),a
        move    x:(r2),y0
        .LOOP   #3
        move    y:(r1+n1),x0
        .IF     a <GT> x0
        move    x0,a
        move    x:(r2+n2),y0
        .ENDI
        move    (r1)+
        move    (r2)+
        .ENDL
        move    a,y:$330
        move    y0,y:$331
        rts

SKRYF_IN_MATRIKS        ;Skryf van wenner sowel as afkomstige pad in matriks.
        move    r6,r0
        move    r7,n0
        move    y:$331,x0
        move    x0,x:(r0+n0)
        move    y:$330,y0
        move    y0,y:(r0+n0)
        move    y0,y:(r1)
        rts

SKRYF_NUWE_IN_OU
        move    #>$310,r1
        .LOOP   #8
        move    y:(r1),y0
        .IF     y0 <GT> #>$FFFFF
        move    #>$310,r2
        .LOOP   #8
        move    y:(r2),a
        asr     a
        asr     a
        move    a,y:(r2)+
        .ENDL
        .ENDI

        move    y:(r1),y0
        move    y0,x:(r1)+
        .ENDL
        rts

KORTSTE_VAN_PAAIE
        move    #>$400,r0
        move    r7,n0
```

```
        move    #>$0,r1
        lua             (r0)+n0,r0
        move    #>$10,n0
        move    r1,y1
        move    y:(r0),a
        .LOOP   #7
        lua     (r0)+n0,r0
        move    (r1)+
        move    y:(r0),x0
        .IF     a <GT> x0
        move    x0,a
        move    r1,y1
        .ENDI
        nop
        .ENDL
        move    y1,y:$370
        rts



;*************************************************************************
;Terugspoor in matriks ten einde wenner te bepaal.
;*************************************************************************

TERUG_SPOOR
        move    #$400,r0
        move    r7,n0
        move    #>15,m0
        move    y:$370,y1
        lua     (r0)+n0,r0
        move    #>$10,x1
        move    #>$361,r3
        .LOOP   #14
        mpy     x1,y1,a
        asr     a
        move    a0,a
        move    r0,x0
        add     x0,a
        move    a,r1
        nop
        move    x:(r1),y1
        move    y1,y:(r3)+
        move    (r0)-
        .ENDL
        move    y1,y:$370
        move    #>$ffff,m0
        rts

dekodeer2
        clr     a
        move    a1,y1
        move    a1,x1
        move    a1,x:dek_ant
        btst    #2,x:ant_0
        jscs    ww1
        btst    #0,x:ant_0
        jscs    ww2
        btst    #0,x:ant
```

```
        jscs    ww4
        move    x1,a1
        eor     y1,a1
        btst    #0,a1
        jscs    ww3
        clr     a
        move    a1,x1
        move    a1,y1
        rts

ww1
        bset    #0,y1
        rts

ww2
        bset    #1,x:dek_ant
        rts

ww3
        bset    #0,x:dek_ant
        rts

ww4
        bset    #0,x1
        rts


;*************************************************************************
;
;Opstelling van geheu vir Viterbidekodering.
;*************************************************************************
;

opstel_v
        move    #>$0,r1
        move    #>$300,n1
        .LOOP   #8
        move    r1,x:(r1+n1)
        move    (r1)+
        .ENDL
        move    #>$0,x0
        move    x0,x:$400
        move    #>$1,x0
        move    x0,x:$410
        move    #>$2,x0
        move    x0,x:$420
        move    #>$3,x0
        move    x0,x:$430
        move    #>$4,x0
        move    x0,x:$440
        move    #>$5,x0
        move    x0,x:$450
        move    #>$6,x0
        move    x0,x:$460
        move    #>$7,x0
        move    x0,x:$470
        move    #>$0,x0
        move    x0,x:$340
        move    #>$2,x0
        move    x0,x:$341
        move    #>$4,x0
        move    x0,x:$342
```

```
        move    #>$6,x0
        move    x0,x:$343
        move    #>$1,x0
        move    x0,x:$344
        move    #>$3,x0
        move    x0,x:$345
        move    #>$5,x0
        move    x0,x:$346
        move    #>$7,x0
        move    x0,x:$347
        rts


;*************************************************************************
;Bepaaling van vierkantswortel.
;*************************************************************************

sqrt
        clr     b       #>1,x1
        move    b,r0
        move    a1,b0
        move    #<2,n1

        do      #12,end1
        asl     b       r0,n0
        asl     b
        tfr     b,a     (r0)+n0
        sub     x1,a    r0,r1
        move    r0,x0
        sub     x0,a    (r1)+n1
        tpl     a,b     r1,r0
        move    r0,a
        asr     a
        if      ROUND==1
        move    #<2,a0
        asr     a
        rnd     a
        endif

end1
        rts

;*************************************************************************
;Bepaling van afstand van ontvangde punt na al die moontlike fases.
;*************************************************************************

afstand
        move    ssh,x:(r6)+
        move    b0,x:(r6)+
        move    b1,x:(r6)+
        move    x0,x:(r6)+
        move    x1,x:(r6)+
        move    y0,x:(r6)+
        move    r1,x:(r6)+
        move    #>6,x0
        move    x0,x:(r6)+
        move    #>14,b
        move    b1,x:(r6)+
        jsr     Felk
        move    (r6)-
```

```
        move    (r6)-
        move    a1,y:Fant0
        move    b1,x:(r6)+
        move    x0,x:(r6)+
        jsr     Felk
        move    (r6)-
        move    (r6)-
        move    a1,y:Fant1
        move    b1,x:(r6)+
        move    #>16777210,x1
        move    x1,x:(r6)+
        jsr     Felk
        move    (r6)-
        move    (r6)-
        move    a1,y:Fant2
        move    x0,x:(r6)+
        move    #>16777202,y0
        move    y0,x:(r6)+
        jsr     Felk
        move    (r6)-
        move    (r6)-
        move    a1,y:Fant3
        move    x1,x:(r6)+
        move    y0,x:(r6)+
        jsr     Felk
        move    (r6)-
        move    (r6)-
        move    a1,y:Fant4
        move    y0,x:(r6)+
        move    x1,x:(r6)+
        jsr     Felk
        move    (r6)-
        move    (r6)-
        move    a1,y:Fant5
        move    y0,x:(r6)+
        move    x0,x:(r6)+
        jsr     Felk
        move    (r6)-
        move    (r6)-
        move    a1,y:Fant6
        move    x1,x:(r6)+
        move     b1,x:(r6)+
        jsr     Felk
        move    (r6)-
        move    (r6)-
        move    a1,y:Fant7

equ
         move           #$300,r1
         clr            a
         clr            b
         move           y:$300,a1
         move           a1,X:klein
         move           (r1)+
          move          y:(r1),a1
         .IF            a1  <LT>  x:klein
          move          a1,x:klein
         .ENDI
         move           (r1)+
          move          y:(r1),a1
```

```
        .IF             a1  <LT>  x:klein
         move           a1,x:klein
        .ENDI
        move            (r1)+
         move           y:(r1),a1
        .IF             a1  <LT>  x:klein
         move           a1,x:klein
        .ENDI
        move            (r1)+
         move           y:(r1),a1
        .IF             a1  <LT>  x:klein
         move           a1,x:klein
        .ENDI
        move            (r1)+
         move           y:(r1),a1
        .IF             a1  <LT>  x:klein
         move           a1,x:klein
        .ENDI
        move            (r1)+
         move           y:(r1),a1
        .IF             a1  <LT>  x:klein
         move           a1,x:klein
        .ENDI
        move            (r1)+
         move           y:(r1),a1
        .IF             a1  <LT>  x:klein
         move           a1,x:klein
        .ENDI

equ1
        .WHILE          a1  <GT>  #>500
         asr            a
         move           #$300,r1
         .LOOP          #8
          clr           b
          move          y:(r1),b1
          asr           b
          move          b1,y:(r1)+
          .ENDL
        .ENDW
        move    (r6)-
        move    x:(r6)-,r1
        move    x:(r6)-,y0
        move    x:(r6)-,x1
        move    x:(r6)-,x0
        move    x:(r6)-,b
        move    x:(r6),b0
        tst     a       (r6)-
        move    x:(r6),ssh
        nop
        rts

Felk
        move    r0,x:(r6)+
        lua     (r6)+,r0
        move    b0,x:(r6)+
        move    b1,x:(r6)+
        move    x0,x:(r6)+
        move    x1,x:(r6)+
        move    y0,x:(r6)+
```

```
        move    y1,x:(r6)+
        move    x:sin_filt,a
        move    #>50,x1
        tfr     a,b
        abs     a
        clr     a       a1,x0
        move    x0,a0
        asl     a
        rep     #$18
        div     x1,a
        eor     x1,b
        jpl     L37
        neg     a
L37
        move    a0,a
        nop
        move    #65533,n0
        nop
        move    x:(r0+n0),y1
        sub     y1,a
        move    a1,x0
        move    x:cos_filt,a
        tfr             a,b
        abs     a
        clr     a       a1,y0
        move    y0,a0
        asl     a
        rep     #$18
        div     x1,a
        eor     x1,b
        jpl     L38
        nop
        neg     a
L38
        move    a0,a
        nop
        move    #65532,n0
        nop
        move    x:(r0+n0),y1
        sub     y1,a
        move    a1,y0
        mpy     y0,y0,a
        mac     +x0,x0,a
        asr     a
        move    a0,a
        move    a1,x:Fantw
        move    (r6)-
        move    x:(r6)-,y1
        move    x:(r6)-,y0
        move    x:(r6)-,x1
        move    x:(r6)-,x0
        move    x:(r6)-,b
        move    x:(r6),b0
        tst     a       x:-(r6),r0
        rts

endd
        end
```

Bronnelys

1 **BATEMAN, A.** Digital Signal Processing Design. London, Pitman. 1988.

2 **COSTELLO, D.J.** Error control coding. Englewood Cliffs, Prentice-Hall. 1988.

3 **GIBSON, J.D.** Principles of digital and analog communications. New York, Macmillan Publishing Company. 1993.

4 **HAYKIN, S.** Digital Communications. New York, John Wiley & Sons.

5 **JERUCHIM, M.C., BALABAN, P. en SHANMUGAN, K. S.** Simulation of Communication Systems. New York, Plenum Press, New York. 1992.

6 **JERVIS, B.W. en Ifeachor, E.C.** Digital Signal Processing. Workingham, England, Addison-Wesley. 1993.

7 **JORDAAN, G.D.** An investigation into synchronization for partial response signals and the development of a clock recovery scheme for 49QPRS signals. Bloemfontein, Ongepubliseerde verhandeling, Technikon Vrystaat. 1997.

8 **LAFRANCE, P.** Fundamental concepts in communication. Englewood-Cliffs, Prentice-Hall. 1990.

9 **LATHI, B.P.** Moderen Digital and Analog Communication Systems, 2de uitgawe. Philadelphia, Holt, Rinehart and Winston, Inc. 1989.

10 MOTOROLA Internet datasheet.

11 **PEEBLES, P.Z.** Digital Communication Systems. Englewood-Cliffs, Prentice-Hall. 1987.

12 **PROAKIS, J.G.** Digital Communications, 2de uitgawe. McGraw-Hill, New York. 1989.

13 **RODDY, D. COOLEN, J.** Electronic Communications, Englewood Cliffs, Prentice-Hall. 1984.

14 **RODEN, M.S.** Digital Communication Systems Design, Englewood Cliffs, Prentice-Hall. 1988.

15 **SCHAERER, T.** A/D and D/A conversion, Elektor Issue No. 83. 1982.

16 **SCHWEBER, W.L.** Data communications,. United States, Mc Graw-Hill. 1988.

17 **SKLAR, B.** Digital Communications, Englewood Cliffs, Prentice-Hall. 1988.

18 **STREMLER, F.G.** Introduction to Communication Systems, 3de uitgawe. Reading, Addison-Wesley, 1990.

19 **TEXAS INSTRUMENTS.** Datablad TLC32044. 1995.

20 **TOMASI, W.** Advanced electronic communications systems. Englewood Cliffs, Prentice-Hall. 1987.

21 **VITERBI, A.J. OMURA, J.K.** Principles of Digital Communication and Coding. Tokyo, McGraw-Hill. 1979.